COMP160: Algorithms, **Homework 8**

- To obtain full credit, you must justify your answers. When describing an algorithm, do not forget to **state any assumptions that you make**, analyze its running time and explain why the algorithm is correct.
- Although not specifically stated, You can assume that we look for algorithms that are as fast as possible, and bounds as tight as possible.
- You may discuss these problems with others, but remember to write the answers on your own. In case of doubt, cite any source you used to do the assignment.
- Remember to submit each question in a separate page.

1. Given an undirected graph in the adjacency list format, how can you decide whether the graph is a tree? How much time will it take, and why? Express the runtime as a function of n (number of vertices) and m (number of edges). What if you are given G in the adjacency matrix format instead?

2. Given a weighted connected graph $G = (V, E)$ and source vertex s, let n be the number of vertices of G, m the number of edges.

   (a) (Warm-up) What is the runtime of the Bellman-Ford algorithm? Justify with 1 sentence

   (b) Let k be the maximum length (measured in number of edges) of all shortest paths (measured in sum of weights) from s in G. For example, if the shortest path in G uses 3 edges, then $k = 3$ (but the *weight* of the shortest path would be the sum of the weights of the three edges). Say that, in addition to G we are also given k as part of the input.

   Modify Bellman-Ford algorithm so that it runs faster. What is the new runtime? Express your answer as a function of k, n and m.

   (c) The algorithm you gave was probably using k somewhere in the algorithm. What happens if k is *unknown*? Can you modify the algorithm so that it still works? If so, what is the new runtime?

3. An infectious disease starts at a node x in a weighted graph. The time it takes to spread along an edge is equal to the weight of the edge. At the same time as the disease is released, we release an antidote at another node $y \neq x$. The antidote moves along the graph just as the disease does.

   Both the antidote and the disease behave in the same way:

   - Initially all nodes (except x and y) are in neutral mode
   - If the infection reaches an neutral node, the disease infects it (forever) and continues spreading to nearby nodes.
   - Similarly, if the antidote reaches an neutral node it protects it (forever), and continues protecting nearby nodes

- If the antidote reaches an infected node, it does nothing (the node remains infected and the antidote does not spread along incident edges from that node).
- Likewise, if the infection reaches a protected node, then it does not spread past that node (and does not change its status).
- If any node is reached simultaneously by both the disease and the antidote, then they neutralize each other: the node is forever *damaged*, and neither gets to keep traveling along incident edges.

Show an algorithm that determines which nodes will be infected, protected, and damaged. Discuss the time and space complexities as well as any additional assumptions you make.

Hint: This is a SSSP problem, but with two sources. You will need to use one of the textbook algorithms explained in class and do some modifications. Make sure to clearly explain those modifications (such as how are priorities handled, how do you determine if the spreading is infection or antidote, and so on). Pay special attention to the handling of ties.

Note: this is a great practice question and the perfect example of the *designing an algorithm* type of question (described in the *how to write perfect proofs* document). Before answering it refresh the document and try to follow that structure. There are a few tricky cases that are easy to miss. Try to give a description that handles ALL cases!