You are allowed to discuss with others but are not allowed to use any references other than the course notes and the reference books. Please list your collaborators for each question. You must write your own solutions. See the course outline for the homework policy.

This homework accounts for 8% of your total grade.

Unless otherwise stated, giving an algorithm means providing a high level description, pseudocode (if needed to clarify the algorithm), a correctness argument, and runtime analysis.

Due date: November 22nd at 11:59pm

Problem 1 (25 Points) - There are a lot of flights flying back and forth between country A and country B. Since it is a high season of flu, both governments would like to cooperate to protect their countries from wide range of infections. The governments plan to install powerful flu detection machines in some airports, such that all passengers are scanned by the machine at least once either at the departure airport or the arrival airport. Since that machine is quite expensive, the governments would like to minimize the number of machines they must install in the airports. Given all the flight information between the two countries, our task is to decide which airports to install the machines.

Input: The airports in country A are called $\{a_1, a_2, ..., a_M\}$ and the airports in country B are called $\{b_1, b_2, ..., b_N\}$. There are F flights between the two countries, where each flight is denoted by (a_i, b_j) for some $1 \leq i \leq M$ and $1 \leq j \leq N$.

Output: The minimum number of airports to install the machines so that for each flight there is a flu machine either at the departure airport of the arrival airport (or both).

Model this as a graph problem and design an efficient algorithm to solve the problem. You will get full marks if the time complexity is polynomial[1] in the unit cost model and the proofs are correct.

[1]Recall polynomial time means the runtime is in $O(s^c)$ where s is the input size and c is a constant.

Problem 2 (25 Points) - You are given an array of n **distinct natural numbers** $A[1..n]$, each containing k digits, where the most significant digit is in $\{1..9\}$. Your goal is to find an optimal path from $A[1]$ to $A[n]$. What do we mean by a path? You can move from $A[i]$ to $A[j]$ if they differ in **one digit**, and you may only move from numbers in A to other numbers in A.

So, for example, if $A = [1234, 4321, 3234, 5274, 3274, 5074]$, you cannot move directly from 1234 to 5074, because they differ in three digits: $1 \neq 5, 2 \neq 0, 4 \neq 7$. Similarly, you cannot move from 1234 to 4321 since they differ in four digits. However, there *is* a path from $A[1]$ to $A[n]$, namely, 1234 to 3234 to 3274 to 5274 to 5074.

What do we mean by optimal? Moving from $A[i]$ to $A[j]$ costs $2A[j] - A[i]$. An optimal path has the smallest total cost. Note that the cost might actually be negative (representing profit). In the example above, the stated path is also optimal.

The output should consist of entries in $A[1..n]$ that form an optimal path from $A[1]$ to $A[n]$. You can return any reasonable data structure (e.g., array/list); Just be clear about what you are doing. If there is no path from $A[1]$ to $A[n]$, you can return IMPOSSIBLE.

You may assume that you can compare two strings, or convert between integer and string representations, in $O(k)$ time. You may call graph algorithms described in class, and you may make (and clearly state) any reasonable input/output assumptions for such algorithms. You can write helper functions if you like. You can use simple data structures like arrays, lists, stacks, queues, etc. Clearly state your assumptions regarding their capabilities.

1. Write an expression for the cost of a path starting and ending at the same number, in terms of A[1..n].

2. Give a high level description and pseudocode (at a higher level than actual code!) showing how to solve this problem using a graph. Be sure to specify whether your graph is directed or undirected, and describe your nodes, edges and weights. Once you've described your nodes, edges and weights, you don't need to explicitly build a graph representation... you can just assume the graph has been built in your preferred representation (but you should state what that representation is!).

3. Argue correctness for your algorithm. In particular, explain why any algorithms you use are applicable to the problem, and why the correct answer is returned. (There exists a two sentence answer warranting full marks.)

4. Analyze the runtime of your algorithm in terms of k and n using big-O notation (unit cost model).

Problem 3 (25 Points) - A transport company operates stations $1, ..., n$, providing customers with the ability to travel between certain stations for a fee. You are given a weighted directed graph $G = (V, E)$, in whichever graph representation you prefer. Each station is represented as a vertex, and there is an edge (u, v) with weight $w(u, v) > 0$ iff it is possible to travel directly from station u to station v. If direct travel from u to v is impossible (so the only way to travel from u to v is to first travel from u to some other station x), then $w(u, v) = \infty$. A trip $T = i_1, i_2, ..., i_\ell$ is a (simple) path in this graph. The cost of trip T is the sum of the edge weights on the path, i.e., $\sum_{j \in \{1...\ell-1\}} w(i_j, i_{j+1})$.

The company wants to run a promotion to bring in more business, and is considering the following. Suppose each customer is given a coupon, which allows the customer to travel for free from one station to another. That is, if a customer's trip T costs $\sum_{j \in \{1...\ell-1\}} w(i_j, i_{j+1})$ dollars, they customer can use their coupon to set $w(i_j, i_{j+1}) = 0$ for **one choice of** j, thereby reducing the cost of their trip. Note the customer can choose which j to use the coupon on.[2]

Assume that any customer taking a trip T from a station s to a station t uses this coupon rationally to plan an optimal trip, meaning the customer pays the smallest possible amount to travel from station s to station t. The company would like to determine, for **all** possible starting and ending stations s and t, the cost that such a rational customer will pay for an optimal trip from s to t (or ∞ if no such trip is possible). The output can be a matrix / 2D array, or any other representation you like.

Provide an efficient solution with the usual correctness and runtime arguments (in the unit cost model).

[2]The coupon cannot be used to traverse an impossible edge with weight ∞.

Problem 4 (25 Points) - Monotonically Increasing Shortest Path

Suppose we are given a weighted directed graph $G = (V, E; w)$ with n vertices and m edges. An edge weight we could be negative, but we assume there is no negative cycle in G. Given a designated vertex $s \in V$, design an algorithm to compute the length of the shortest monotonically increasing path from s to every other vertex $v \neq s$. A path is monotonically increasing if the weight of every edge on the path is strictly increasing.

Remark: your algorithm should run in $O(n^2 m)$ (unit cost model) to get full marks.

Hint: try some variant of Bellman-Ford algorithm.