CONCORDIA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

COMP 6651/4: Algorithm Design Techniques

Fall 2015

MidTerm - Close book exam - 3 hours

Instructor: Professor B. Jaumard
Office: EV 003.115
Email: bjaumard@cse.concordia.ca

First Name	Last Name	ID#

- Whenever you need to use one of the algorithms presented in the lecture slides, you need to recall the details of that algorithm.
- Whenever you are asked to design an algorithm, you need to provide a detailed pseudo-code, as well as the definition of all the parameters/variables used in the algorithm.
- Whenever you are asked to design an efficient algorithm, you need to provide the best algorithm you can think of. Points will still be given for an algorithm that is not the best one, as far as it is a correct one, and if it is not too naive/simple. However, it is better to propose a simple algorithm than no algorithm.

Question 1. (20 points.)

An extendable array is a data structure that stores a sequence of items and supports the following operations.

- AddToFront(x) adds x to the beginning of the sequence.
- AddToEnd(x) adds x to the end of the sequence.
- LookUp(k) returns the kth item in the sequence, or NULL if the current length of the sequence is less than k.

Describe a simple data structure that implements an extendable array. Your AddToFront and AddToBack algorithms should take $O(1)$ amortized time, and your LOOKUP algorithm should take $O(1)$ worst-case time. The data structure should use $O(n)$ space, where n is the current length of the sequence.

Question 2. (20 points.)

Suppose you are given three strings $A[1..n]$, $B[1..n]$, and $C[1..n]$. Describe and analyze an algorithm to find the maximum length of a common subsequence of all three strings. For example, given the input strings

$$A = \textbf{AxxBxxCDxEF}, \qquad B = \textbf{yyABCDyEyFy}, \qquad C = \textbf{zAzzBCDzEFz},$$

your algorithm should output the number 6, which is the length of the longest common subsequence **ABCDEF**.

Question 3. (20 points)

In machine learning applications, we often have some kind of condition defined over a set, S, of n points, which we would like to characterize - that is, learn - using some simple rule. For instance, these points could correspond to biological attributes of n medical patients and the condition could be whether a given patient tests positive for a given disease or not, and we would like to learn the correlation between these attributes and this disease. Think of the points with positive tests as painted red, and the tests with negative tests as painted blue. Suppose that we have a simple two-factor set of attributes; hence, we can view each patient as a two-dimensional point in the plane, which is colored as either red or blue. An ideal characterization, from a machine learning perspective, is if we can separate the points of S by a line, ℓ, such that all the points on one side of ℓ are red and all the points on the other side of ℓ are blue. So suppose you are given a set, S, of n red and blue points in the plane. Describe an efficient algorithm for determining whether there is a line, ℓ, that separates the red and blue points in S. What is the running time of your algorithm?

Question 4. (20 points)

Let $G = (V, E)$ be a directed graph with edge capacities $c : E \to \mathbb{R}^+$, a source vertex s, and a target vertex t. Suppose someone hands you an arbitrary function $f : E \to \mathbb{R}$. Describe and analyze fast and simple algorithms to answer the following questions:

(a) Is f a feasible (s, t)-flow in G?

(b) Is f a maximum (s, t)-flow in G?

(c) Is f the unique maximum (s, t)-flow in G?

Question 5. Social network connectivity. (20 points)

Given a social network containing N members and a log file containing M timestamps at which times pairs of members formed friendships, design an algorithm to determine the earliest time at which all members are connected (i.e., every member is a friend of a friend of a friend of a friend). Assume that the log file is sorted by timestamp and that friendship is an equivalence relation. The running time of your algorithm should be $O(M \log N)$ or better and use extra space proportional to N.