University of Toronto, Faculty of Arts and Science
August 2016 Examinations
CSC373H1Y Final Examinations
Lalla Mouatadid
Duration: 3 hours
No Aids Allowed

PLEASE COMPLETE THE SECTION BELOW:

First Name: ____________________________________

Last Name: ____________________________________

Exam Instructions

- **Check that your exam book has 17 pages** (including this cover page and 3 blank pages at the end). The last 3 pages are for rough work only, *they will* **not** *be marked.* Please bring any discrepancy to the attention of an invigilator.
- There are 7 questions worth a total of 80 points. Answer all questions on the question booklet.
- For Questions F and G if you do not know how to answer any part of the question then you can leave that part blank or write "I DON'T KNOW." to receive 20 percent of the points of the question.
- **You must get at least 40% on this exam in order to pass the course.**

Course Specific Notes

- Unless stated otherwise, you can use the standard data structures and algorithms discussed in CSC263 and in the lectures without describing their implementation by simply stating their standard name (e.g. min-heap, merge- sort, DFS, Dijkstra). Suggest a one must-do activity in Toronto for one extra bonus mark. You do not need to provide any explanation or pseudo-code for the data structures implementation. You can also use their running time without proof. For example, if you are using the merge-sort in your algorithm you can simply state that merge-sort's worst-case running time is $\mathcal{O}(n \log n)$. If you modify a data structure or an algorithm from class, you must describe the modification and its effects.
- In some questions you will be given a computational problem and then asked to design an efficient algorithm for it. Unless stated otherwise, for data structures and algorithms that you design you should provide a short high-level explanation of how your algorithm works in plain English, and the pseudo-code for your algorithm in a style similar to those we have seen in the lectures. If you miss any of these the answer might not be marked. Your answers will be marked based on the efficiency of your algorithms and the clarity of your explanations. State the running time of your algorithm with a brief argument supporting your claim and prove that your algorithm works correctly (e.g. finds an optimal solution).

PLEASE PRINT YOUR STUDENT NUMBER AND YOUR NAME

Student Number: ________________________

First Name: ________________________

Last Name: ________________________

The section below is for marker's use only. Do NOT use it for answering or as scratch paper.

Questions	**Points**
A	/10
B	/6
C	/10
D	/11
E	/8
F	/20
G	/15
Total	/80

A. True/False [10]

Circle True or False, do not justify your answers.

1. **True** or **False**: If $G(V, E)$ is an undirected graph where $|E| = |V| - 1$, then G is guaranteed to be a tree.

2. **True** or **False**: Let $G(V, E, w)$ be an edge-weighted graph, with $w : E \rightarrow \mathbb{R}^{>0}$, and s, t two vertices in V. If all the edge weights are distinct, then there is a unique shortest path from s to t.

3. **True** or **False**: If all of the edge capacities in a graph are an integer multiple of 7, then the value of the maximum flow will be a multiple of 7.

4. **True** or **False**: If we want to prove that a search problem X is NP-complete, it's enough to reduce 3-SAT to X (in other words, it's enough to prove 3-SAT $\leq_p X$).

5. **True** or **False**: If we want to prove that a search problem X is NP-complete, it's enough to reduce X to 3-SAT (in other words, it's enough to prove $X \leq_p$ 3-SAT).

6. **True** or **False**: If problem A can be reduced to problem B, and problem B can be reduced to problem C, then A can also be reduced to C.

7. **True** or **False**: We can reduce the search problem MAXIMUM FLOW to the search problem LINEAR PROGRAMMING (in other words, MAXIMUM FLOW $\leq_p$ LINEAR PROGRAMMING).

8. **True** or **False**: If X is any search problem, then X can be reduced to INDEPENDENT SET.

B. Short Answer [6]

Answer the following questions, giving a short justification (a sentence or two).

1. If P $\neq$ NP, could there be a polynomial time algorithm for 3-SAT?

2. Let $G(V, E, w)$ be an undirected, connected, edge-weighted graph where $w(e) > 0, \forall e \in E$. Let T be a minimum spanning tree of G. Suppose we modify the graph by adding 9 to the weight of each edge. Is T still a minimum spanning tree of the modified graph?

C. Tiny Changes. Big Costs. [10]

Consider two problems, Short Path and Long Path. Both take as input a graph $G(V, E)$, two specific vertices u and v and an integer k. Long Path asks whether there is a path in G from u to v of length *at least* k, and Short Path asks whether there is a path in G from u to v of length *at most* k. In both cases, paths are not allowed to visit the same vertex twice.

One of these problems is in P and the other one is NP-hard.

1. Clearly specify which problem is in P, and give an efficient algorithm to solve it (with a short description). No need to prove correctness. [5]

2. Clearly specify which problem is NP-hard by giving a reduction from an NP-hard problem of your choice. Give a short description as to why your reduction works. [5]

D. Canoes are the future... [11]

You've been hired by Canuth, a new startup with this great idea to make a killing by creating a market for canoes futures. Canuth has relationships with a set of n suppliers and a set of m purchasers. The i^{th} supplier can supply up to $s[i]$ canoes this year, and the j^{th} purchaser would like to buy up to $b[j]$ canoes this year. Canuth is the middleman and makes \$1 off each canoe that is sold. Thus, the more canoes that are sold, the more money you make!

However, there is complication. Due to federal restrictions on interstate trafficking in outdoor gear, supplier i can only sell canoes to a purchaser j if the are situated at most 100 miles apart. Assume that you're given a list L of all pairs (i, j) such that supplier i is within 100 miles of purchaser j. You will be given $n, m, s[1, \ldots, n], b[1, \ldots, m], L$ as input. Your job will be to compute the maximum number of canoes that can be sold this year.

1. Formulate this as a network flow problem. Clearly explain how you construct your network, and why the network captures the problem. [5]

2. Formulate this as a linear programming problem. Clearly explain what every variable and constraint is for. [5]

3. Given the fact that all your variables/input are integral, and suppose you don't care about the running time of your algorithm; which formulation would be better, network flow or linear programming? [1]

E. Two Birds. One Stone[1] [8]

Suppose you were given an approximation algorithm A for the independent set problem, that has an approximation ratio bounded by two. Describe how you can use A to obtain a 2-approximation algorithm for the clique problem.

[1]Unfortunately there is no 2-approximation algorithm for Independent Set.

F. Dynamic Programming [20]

Suppose that you are consulting for a company that manufactures and ships PC equipment to distributors around the country. For each of the next n weeks, they have a projected *supply* s_i of equipment (measured in pounds), which has to be shipped by an air freight carrier.

Each week's supply can be carried by one of two air freight companies, A or B.

- Company A charges a fixed rate r per pound (so it costs $r \cdot s_i$ to ship a week's supply s_i).
- Company B makes contracts for a fixed amount c per week, independent of the weight. However, contracts with company B must be made in blocks of four consecutive weeks at a time.

A *schedule*, for the PC company, is a choice of air freight company (A or B) for each of the n weeks, with the restriction that company B, whenever it is chosen, must be chosen for blocks of four contiguous weeks at a time. The *cost* of the schedule is the total amount paid to company A and B, according to the description above.

Give a polynomial-time dynamic programming algorithm that takes a sequence of supply values $s_1, s_2, \ldots, s_n$ and returns a *schedule* of minimum cost.

Example: Suppose $r = 1, c = 10$, and the sequence of supply values is

$$11, 2, 2, 12, 12, 12, 12, 9, 9, 11.$$

Then the optimal schedule would be to choose company A for the first three weeks, then company B for a block of four consecutive weeks, and then company A for the final three weeks.

1. Give a high-level description of each cell in the recurrence used by your algorithm. [3]

2. Formally define the recurrence relation for the recurrence you gave in Part 1. [4]

3. Give a dynamic programming algorithm implementing the recurrence relation. Be sure to give a brief high-level description of your algorithm, and a pseudo-code implementation. [6]

4. Prove that the recurrence relation you defined is correct. [7]

G. I found this one! No I did! [15]

You and a friend have been trekking through various far-off parts of the world and have accumulated a big pile of souvenirs. At the time you weren't really thinking about which of these you were planning to keep and which your friend was going to keep, but now the time has come to divide everything up.

Here's a way you could go about doing this. Suppose there are n objects, labeled $1, 2, \dots, n$, and object i has an agreed-upon value x_i. One reasonable way to divide things would be to look for a *partition* of the objects into two sets, so that the total value of the objects in each set is the same.

This suggest solving the following *Number Partitioning Problem.* You are given positive integers $x_1, \dots, x_n$; you want to decide whether the numbers can partitioned into two sets S_1, and S_2 with the same sum:

$$\sum_{x_i \in S_1} x_i = \sum_{x_j \in S_2} x_j$$

Show that Number Partitioning is NP-complete.[2]

[2]It was the end of a good friendship :(

Extra page for Question G - if needed.

Will not be graded.