UNIVERSITY OF TORONTO
Department of Computer Science

April 2014 Final Exam

CSC373H1S

Robert Robere

Duration - 3 hours

No Aids Allowed.

PLEASE COMPLETE THE SECTION BELOW AND THE SECTION BEHIND THIS PAGE:

First Name: _____

Last Name: _____

Exam Instructions

- **Check that your exam book has 19 pages** (including this cover page and 4 blank pages at the end). The last 4 pages are for rough work only, *they will* **not** *be marked.* Please bring any discrepancy to the attention of an invigilator.
- There are 8 questions worth a total of 115 points. Answer all questions on the question booklet.
- For Questions 4, 5, 6, and 7, if you do not know how to answer a part of the question then you can leave that part blank or write "I DON'T KNOW." to receive 20 percent of the points for that part.
- **To pass this course you must achieve at least** 40% **on this final exam.**

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). You do not need to provide any explanation or pseudo-code for their 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 $O(n \log n)$. 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. If you draw a picture of a spooky monster on the back of this exam you will receive one extra bonus mark. 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 (i.e. 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.

Question #	Score/Points
1	/ 12
2	/ 7
3	/ 12
4	/ 4
5	/ 20
6	/ 20
7	/ 20
8	/ 20
Total	/ 115

1. **Problem Definitions** [12]

Define the inputs and outputs of each of the following problems.

a. **All-Pairs Shortest Paths** [3]

Input:

Output:

b. **Independent Set** [3]

Input:

Output:

c. **Circuit SAT** [3]

Input:

Output:

d. **Matrix Chain Multiplication** [3]

Input:

Output:

2. **Algorithm Definitions** [7]

a. State the names of two algorithms studied in this class that solve the single-source shortest paths problem. [2]

b. What distinguishes each of these algorithms from one another? [2]

c. State the Max-Flow/Min-Cut theorem, and briefly describe at a high level what it means. [3]

3. **Complexity Theory** Give a formal definition for each of the following topics/mathematical objects that we examined in complexity theory. [12]

   a. **Many-to-one reductions** [3]

   b. NP [3]

   c. NP-**Hardness** [3]

   d. **Search-to-decision reductions** [3]

4. **Algorithmic Paradigms** [4]

   As stated in class and in the lecture notes, what are the two distinguishing features (or "hallmarks") of dynamic programming algorithms?

5. **Frugal Flying** [20]

You are a helicopter pilot for the FIRST-ACE helicopter company, and you have just been assigned to a new contract. The details of the contract are kept hidden (for super-secret security purposes), but you know this: you have been given a particular road to monitor (of length ℓ, for some positive integer ℓ). Every hour you will be sent a list of *non-intersecting* intervals on the road that you need to monitor (that is, a list of integer pairs $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$, where $I_j = [s_{I_j}, f_{I_j}] \in \mathbb{Z}^2$, with $1 \leq s_I < f_I \leq \ell$ for all I) with a subtly-installed spy-cam on the bottom of your helicopter. Each pair $[s_{I_j}, f_{I_j}]$ represents a set of integer points on the road $\left\{s_{I_j}, s_{I_j}+1, s_{I_j}+2, \ldots, f_{I_j}-1, f_{I_J}\right\}$ that you need to film with your camera. **You may assume that the intervals are sorted in increasing order by start time s_I.**

Unfortunately, due to budget cuts, your spy-cam only has a limited amount of film, and once it turns on it can never be turned off. In particular, it can only film a stretch of the road of length $f \in \mathbb{Z}$. You can, however, choose when to turn it on — for example, if you choose to turn on the spy-cam at point x on the road, then it will film all of the integer points

$$\{x, x+1, x+2, \ldots, x+f-1\}.$$

When the spy-cam turns off you have to return to the start to re-load your film. Before you start to fly, then, your job is to figure out "start points" for filming on the road so that you can cover all of the input road-intervals $\mathcal{I}$ while minimizing the number of flights used.

Input: A positive integer $\ell \in \mathbb{Z}$ representing the length of the road, a positive integer f representing the length of film your spy-camera takes, and a list of intervals $\mathcal{I}$ where for all $I = [s_I, f_I] \in \mathcal{I}$ we have $1 \leq s_I < f_I \leq \ell$.

Output: The minimum number of film reloads needed to cover all of the points in the input intervals. **Note that you do not need to return the actual filming points.**

a. Give a greedy algorithm solving the Frugal Flying problem. Assuming that k is the number of points in the longest interval in the input, then to receive full marks your algorithm must run in $O(nk/f)$ time. Be sure to give a brief, high-level description of your algorithm, a pseudocode implementation, and an argument about the running time. [10]

b. Prove that your greedy algorithm is optimal. [10]

6. **Smile!** [20]

In this problem you will follow one of your deepest and most secret dreams: being a camera person at the Super Bowl! This year, the Super Bowl is being held in the Circleville Circle Dome (one of those famous circular football fields you hear about all the time). You are in charge of capturing any interesting, bizarre, or otherwise remarkable events which happens in the bleachers of the stadium in between the ball-foot action.

To do this, you will be seated in a rotating camera booth in the exact center of the Circle Dome. The camera can rotate to the left or right at a rate of ϕ radians per second. (Assume that when you begin the camera is oriented at 0 radians). From the announcer booth you will receive a list of n *events*, where a single event is defined by a pair $E_i = (\rho_i, t_i)$, where $0 \leq \rho_i < 2\pi$ is the angle (calculated counter-clockwise from your starting orientation at 0 radians) at which the ith event is happening, and t_i tells you when you need to take a picture of the event. To take a picture of the ith event, your camera booth must be at the angle of ρ_i (counter-clockwise) from 0 at time t_i.

In this problem, given a list of n events $\mathcal{L} = \{E_1, E_2, \ldots, E_n\}$, you must return the maximum number of events that you can film.

Camera Booth

Input: A positive real number ϕ, and a list of n events $\mathcal{L} = \{E_1, E_2, \ldots, E_n\}$, as defined above.

Output: The maximum number of events that you can capture if your camera booth starts facing at 0 radians and it can rotate clockwise or counter-clockwise at a speed of ϕ radians per second.

Give a dynamic programming algorithm for this problem. To receive full marks it must run in $O(n^2)$ time.

a. Give a high-level description of each cell in the recurrence used by your algorithm. [4]

b. Formally define the recurrence relation for the recurrence you gave in Part (a). [6]

c. Give a dynamic programming algorithm implementing the recurrence relation. Be sure to give a brief, high-level description of your algorithm, a pseudocode implementation, and an argument about the running time. [5]

d. Prove that the recurrence relation you defined in Parts (a) and (b) is correct. [5]

7. **Vertex Cover with Edge Costs** [20]

Consider the following generalization of the vertex cover problem. You are given an undirected graph $G = (V, E)$, along with positive *vertex costs* w_u for all $u \in V$ and positive *edge costs* c_{uv} for all edges $(u, v) \in E$. As before, you want to find a subset of vertices $S \subseteq V$ so that every edge in the graph G is covered (i.e. has at least one of its vertices) in the set S. However, you may also leave some edges *uncovered*, but for every edge (u, v) left uncovered by a subset of vertices S, you must pay a penalty of c_{uv} for that edge. The *cost* of a subset of vertices S will be the sum of the weights of all of the vertices in S, plus the cost of edges not covered by S:

$$cost(S) = \sum_{u \in S} w_u + \sum_{\substack{(u,v) \in S \\ u,v \notin S}} c_{uv}.$$

Weighted Vertex Cover with Edge Penalties

Input: An undirected graph $G = (V, E)$. Positive real vertex weights w_u for each vertex $u \in V$ and positive real edge costs c_{uv} for each edge $(u, v) \in E$.

Output: A set of vertices $S \subseteq V$ such that

$$cost(S) = \sum_{u \in S} w_u + \sum_{\substack{(u,v) \in E \\ u,v \notin S}} c_{uv}$$

is minimized.

a. Give a $\{0, 1\}$-integer programming formulation for this problem. [10]

b. What constraints would have to be changed in order to find the linear programming relaxation of the integer program in Part (a)? [2]

c. Give a rounding scheme for the linear programming relaxation of your integer program from Part (a). (That is, provide a method of rounding the variables in a solution to the relaxed linear program to a solution of the integer program from Part (a).) Prove that the rounded solution will be a solution to the integer program, and also show that the rounded solution gives a bounded approximation ratio for the Weighted Vertex Cover with Edge Penalties problem. To receive full marks, your rounding scheme must attain a 4-approximation. [8]

8. **Clique** [20]

Recall that if $G = (V, E)$ is an undirected graph, then a *k-clique* is a subset of k vertices $C \subseteq V$ such that for all $u, v \in C$ there is an undiricted edge $(u, v) \in E$.

Clique

Input: A positive integer k, and an undirected graph $G = (V, E)$.

Output: Output 1 if the graph G contains a clique with at least k vertices.

a. Prove that the Clique problem is in NP, by giving a polynomial time many-one reduction to some NP-Complete problem (other than the Clique problem, of course). [8]

b. Prove that the Clique problem is NP-Hard by a reduction from the 3-SAT problem. [12]

This page is for rough work only, it will **not** be graded.

More space for rough work is available on the next page.

This page is for rough work only, it will **not** be graded.

More space for rough work is available on the next page.

This page is for rough work only, it will **not** be graded.

More space for rough work is available on the next page.

This page is for rough work only, it will **not** be graded.