PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

AUGUST 2015 EXAMINATIONS

CSC 373 H1Y
Instructor(s): J. Girard

Duration — 3 hours

Examination Aids: ***One*** **8.5" × 11" sheet of paper,** ***hand*****written on both sides.**

Student Number: ___________

Last (Family) Name(s): ___________

First (Given) Name(s): ___________

Do **not** *turn this page until you have received the signal to start.*
In the meantime, please read the instructions below *carefully.*

This final examination paper consists of 6 questions on 23 pages (including this one), printed on one side of each sheet. *When you receive the signal to start, please make sure that your copy of the final examination is complete, fill in the identification section above, and write your student number where indicated at the bottom of every page (except page 1). You must receive at least 40% on this exam to pass this course.*

Answer each question directly on this paper, in the space provided, and use the reverse side of the pages for rough work. If you need more space for one of your solutions, use the reverse side of a page and *indicate clearly the part of your work that should be marked.*

In your answers, you may use without proof any result or theorem covered in lectures, tutorials, homework, tests, or the textbook, as long as you give a clear statement of the result(s)/theorem(s) you are using. You must justify all other facts required for your solutions.

Write up your solutions carefully! In particular, use notation and terminology correctly and explain what you are trying to do — part marks *will* be given for showing that you know the general structure of an answer, even if your solution is incomplete.

If you are unable to answer a question (or part), you will get 20% of the marks for that question (or part) if you write "I don't know" and nothing else — you will *not* get those marks if your answer is completely blank, or if it contains contradictory statements (such as "I don't know" followed or preceded by parts of a solution that have not been crossed off). "I don't know" is not eligible for bonus marks.

MARKING GUIDE

# 1:	______/15
# 2:	______/10
# 3:	______/10
# 4:	______/15
# 5:	______/15
# 6:	______/10
TOTAL:	______/75

Good Luck!

Question 1. (P and NP, Reducibility) [15 MARKS]

Part (a) [6 MARKS]

For each decision problem D below, state whether $D \in P$, $D \in NP$ or $D \in coNP$ and justify each of your claims briefly:

- for decision problems in P, describe an algorithm to solve the problem and briefly argue that your algorithm is correct and runs in polytime;
- for decision problems in NP or $coNP$, describe a suitable verifier for the problem and briefly argue that your verifier is correct and runs in polytime.

Make the strongest claims that you can.

(i) SMALLSUM [2 MARKS]:

- **Input:** A non-empty finite set of positive integers $S = \{x_1, \ldots, x_n\} \subset \mathbb{Z}^+$, and a positive integer $t \in \mathbb{Z}^+$ (all integers represented in binary).
- **Output:** Does **some** non-empty subset of S have sum **at most** t? ($\exists S' \subseteq S, S' \neq \varnothing \wedge \sum_{x \in S'} x \leqslant t$?)

(ii) LARGESUMS [2 MARKS]:

- **Input:** A non-empty finite set of positive integers $S = \{x_1, \ldots, x_n\} \subset \mathbb{Z}^+$, and a positive integer $t \in \mathbb{Z}^+$ (all integers represented in binary).
- **Output:** Does **every** non-empty subset of S have sum **at least** t?
  $(\forall S' \subseteq S, S' \neq \varnothing \implies \sum_{x \in S'} x \geqslant t?)$

(iii) ExactSum [2 MARKS]:

- **Input:** A non-empty finite set of positive integers $S = \{x_1, \dots, x_n\} \subset \mathbb{Z}^+$, and a positive integer $t \in \mathbb{Z}^+$ (all integers represented in binary).
- **Output:** Does **some** non-empty subset of S have sum **exactly** t?
  ($\exists S' \subseteq S, S' \neq \varnothing \wedge \sum_{x \in S'} x = t$?)

Part (b) [9 MARKS]

Consider the following "PARTITION" search problem.

Input: A set of integers $S = \{x_1, x_2, \ldots, x_n\}$ — each integer can be positive, negative, or zero.

Output: A partition of S into subsets S_1, S_2 with equal sum, if such a partition is possible; otherwise, return the special value NIL. (S_1, S_2 is a "partition" of S if every element of S belongs to one of S_1 or S_2, but not to both.)

(i) [3 MARKS] Give a *precise* definition for a decision problem "PART" related to the PARTITION search problem. Consider a set of inputs and a set of outputs.

(ii) [6 MARKS] Write an algorithm to solve the PARTITION search problem: (Take the time to define your variables and objective. Hint: You likely only want to consider each number x_i once, and for each valid partition x_i must be part of one of two sums.)

Question 1. (CONTINUED)

(iii) **BONUS** [5 MARKS]
BONUS: Do not attempt until all non-bonus questions are completed. This question is optional. Describe the correctness and runtime of your algorithm.

[More space on the next page...]

Question 2. (Network Flow) [10 MARKS]

Consider a flow network in which each vertex has a capacity, in addition to each edge, *i.e.*, a network $N = (V, E)$ with a single source $s \in V$, a single sink $t \in V$, edge capacities $c(e) \geq 0$ for all $e \in E$ and vertex capacities $c(v) \geq 0$ for all $v \in V$ (including s and t).

For such a network, the capacity constraint is extended to include vertices: for each vertex $v \in V$, $f^{\text{in}}(v)$, $f^{\text{out}}(v) \leq c(v)$, *i.e.*, the total flow into or out of any vertex cannot exceed the capacity of that vertex.

Explain how to determine a flow with maximum value in such a network, and justify that your algorithm works properly.

Part (a) [6 MARKS]

Explain construction of your network.

Question 2. (CONTINUED)

Part (b) [4 MARKS]

Explain the runtime and correctness of the solution.

Question 3. (Linear Programming) [10 MARKS]

The **Partially Conserved Maximum Flow** problem is similar to the maximum flow problem, except that:

- each vertex $u \in V - \{s, t\}$ has "conservation factors" $\alpha_u, \beta_u \in \mathbb{R}$ with $0 \leqslant \alpha_u \leqslant \beta_u$;
- for each vertex $u \in V - \{s, t\}$, $\alpha_u f^{\text{in}(u)} \leqslant f^{\text{out}(u)} \leqslant \beta_u f^{\text{in}(u)}$: the total flow out of u is between α_u and β_u times the total flow into u.

As before, individual edge flows must satisfy the capacity constraint ($0 \leqslant f(e) \leqslant c(e)$ for all $e \in E$) and we are looking for an assignment of flow values to every edge that maximizes $f^{\text{out}(s)}$.

Explain how to solve the partially conserved maximum flow problem using linear programming. Give an efficient pseudo-code algorithm that takes the network and constructs the linear programming problem. Include a brief English description of the main idea of your algorithm, justify that your algorithm is correct, and analyze its running time.

Part (a) [5 MARKS]

Explain construction of the linear program.

Part (b) [5 MARKS]

Explain correctness and runtime of solution.

Question 4. (Dynamic Programming) [15 MARKS]

Consider the following problem, which we will call POWERGRID.

Input: A directed acyclic graph $G = (V, E)$, as in the example below. That is, the graph can be laid out on a grid so that all directed edges point either right or down, and no two edges cross. Further assume that $V = \{p_1, \dots, p_n\}$ and that the vertices appear in order on the grid, when scanning left-to-right and top-to-bottom.

Output: The path that contains the maximum number of junction direction changes, connecting p_1 (the vertex at the top left corner) to p_n (the vertex in the lower right). This is seen as a measure of power grid complexity.

For example, in the diagram below, there are three distinct paths from p_1 to p_{10}, ($p_1 \to p_2 \to p_5 \to p_6 \to p_{10}$ with a direction change number of 3, $p_1 \to p_2 \to p_5 \to p_9 \to p_{10}$ with a direction change number of 2, $p_1 \to p_8 \to p_9 \to p_{10}$ with a direction change number of 1). This means p_{10} has a maximum direction change number of 3. There is only one path from p_1 to p_{13}, with a direction change number of 0. ($p_1 \to p_8 \to p_{13}$).

$$\begin{tikzcd}
p_1 \arrow[r] \arrow[dd] & p_2 \arrow[rr] \arrow[d] & & p_3 \arrow[rr] \arrow[d] & & p_4 \arrow[ddd] \\
& p_5 \arrow[r] \arrow[d] & p_6 \arrow[r] \arrow[d] & p_7 \arrow[dd] & & \\
p_8 \arrow[r] \arrow[dd] & p_9 \arrow[r] \arrow[dd] & p_{10} \arrow[dd] & & & \\
& & & p_{11} \arrow[rr] \arrow[d] & & p_{12} \arrow[d] \\
p_{13} \arrow[r] \arrow[d] & p_{14} \arrow[r] \arrow[d] & p_{15} \arrow[r] \arrow[d] & p_{16} \arrow[r] & p_{17} \arrow[r] & p_{18} \arrow[d] \\
p_{19} \arrow[r] & p_{20} \arrow[r] & p_{21} \arrow[rrr] & & & p_{22}
\end{tikzcd}$$

Use the dynamic programming paradigm covered in lecture to solve the POWERGRID problem. Take the time to explain carefully what you are doing and to write up a detailed answer in particular, follow closely the paradigm given in lecture.

Assume that you have at your disposal two arrays $up[i]$ and $left[i]$, defined as follows for $1 \leqslant i \leqslant n$:

$$up[i] = \begin{cases} j & \text{if there is a vertex } x_j \text{ such that } (x_j, x_i) \text{ is an edge going down,} \\ 0 & \text{otherwise.} \end{cases} \qquad left[i] = \begin{cases} j & \text{if there is a vertex } x_j \text{ such that } (x_j, x_i) \text{ is an edge going right,} \\ 0 & \text{otherwise.} \end{cases}$$

For example, in the graph above, $up[3] = 0$ and $left[3] = 2$, $up[11] = 7$ and $left[11] = 0$.

Part (a) [2 MARKS]

Describe the Optimal Substructure and Overlapping Subproblems for this problem.

Part (b) [3 MARKS]
Define the Array or Table needed to store the values.

Part (c) [5 MARKS]

Define the value based recurrence relation used to find an optimal solution.

Part (d) [5 MARKS]

Describe the iterative algorithm needed for this problem.

Part (e) BONUS [5 MARKS]

BONUS: Do not attempt until all non-bonus questions are completed. This question is optional. Describe how to reconstruct the path from your iterative algorithm.

Question 5. (Greedy Algorithms) [15 MARKS]

A busy web server receives many document requests at the same time, and must decide in what order to reply to the requests. Each request takes a certain time for the server to process, and the server can only process one request at a time. But each request also takes a certain time for the web clients to render, independently of the other web clients. Suppose that we are interested in responding to the requests in an order that guarantees all requests will be fully rendered as quickly as possible.

Formally, *consider the following "Web Request Problem"*:

Input: Requests $R_1, R_2, \ldots, R_n$, where each request R_i consists of "server" time s_i and "client" time c_i (both of them positive integers). All server time takes place on a single processor, sequentially. All client time takes place on n independent processors, in parallel.

Output: An ordering of the requests $i_1, i_2, \ldots, i_n$ (a permutation of $[1, 2, \ldots, n]$) that minimizes the overall completion time: $\max\left\{s_{i_1} + s_{i_2} + \cdots + s_{i_k} + c_{i_k} : 1 \leqslant k \leqslant n\right\}$ — where the completion time for request number k is equal to the total server time for all requests processed before k $(s_{i_1} + s_{i_2} + \cdots + s_{i_{k-1}})$, plus the time to fully process and render request k, $(s_{i_k} + c_{i_k})$.

HINT: When you order a request, consider its "end time" $T_k(S) = (s_{i_1} + s_{i_2} + \cdots + s_{i_k} + c_{i_k})$, where $S = \{i_1, i_2, i_3 \ldots i_k\}$.

Part (a) [4 MARKS]

Write a greedy algorithm to solve the *"Web Request Problem"*. You must give a detailed pseudo-code implementation of your algorithm, as well as a high-level English description of the main steps in your algorithm.

Part (b) [3 MARKS]

Give a precise definition of what constitutes a "partial solution" for your algorithm, and what it means for a partial solution to be "promising".

Question 5. (CONTINUED)

Part (c) [8 MARKS]

Give a detailed proof that your greedy algorithm will always return an optimal permutation of requests that prioritizes completion time.

(Your proof will be marked on its structure as well as its content.)

Question 6. (Approximation Algorithms) [10 MARKS]

Consider the Maximum Independent Set problem without weights, *i.e.*, given an undirected graph $G = (V, E)$, find an independent set of G of maximum size.

Write a greedy algorithm that attempts to find a large independent set in its input graph G (include a short English description of the main idea of your algorithm). Then, give a good bound on the approximation ratio for your algorithm when it is run on graphs with maximum degree k (*i.e.*, where each vertex has at most k neighbours), and prove that your bound is correct.

Part (a) [5 MARKS]

Create a greedy algorithm that approximates a solution.

Part (b) [5 MARKS]

What would be a good approximation bound? Briefly discuss.

Total Marks = 75