UNIVERSITY OF TORONTO
Faculty of Arts & Science

December 2019 Examinations

CSC 373 H1F

Duration: 3 hours

Aids Allowed: One *single*-sided *hand*written *8.5"×11"* aid sheet.

*Do **not** turn this page until you have received the signal to start.* In the meantime, write your name, student number, and UTORid below (please do this now!) and *carefully* read *all* the information on the rest of this page.

First (Given) Name(s):

Last (Family) Name(s):

10-Digit Student Number:

UTORid (e.g., `pitfra12`):

- This final examination consists of 7 questions on 16 pages (including this one), printed on both sides of the paper. *When you receive the signal to start, please make sure that your copy of the examination is complete.*
- Answer each question directly on the examination paper, in the space provided, and use a "blank" page for rough work. If you need more space for one of your solutions, use one of the "blank" pages and *indicate clearly the part of your work that should be marked.*
- *Remember that, in order to pass the course, you must achieve a grade of at least **40%** on this final examination.*
- As a student, you help create a fair and inclusive writing environment. If you possess an unauthorized aid during an exam, you may be charged with an academic offence.
- Remember that for any question or subquestion, writing "I do not know how to approach this" gives you 20% of the marks (not writing this still gives you 10% of the marks). However, for this you need to make sure that you scratch off any other text you have written for that subquestion.

MARKING GUIDE

Nº 1:	_____/ 15
Nº 2:	_____/ 20
Nº 3:	_____/ 15
Nº 4:	_____/ 10
Nº 5:	_____/ 15
Nº 6:	_____/ 10
Nº 7:	_____/ 15
BONUS:	_____/ 10
TOTAL:	_____/100

students must hand in all examination materials at the end

December 2019 Examinations

CSC 373 H1F

Duration: **3 hours**

Question 1. Greedy [15 marks]

There are n boxes arranged in a row. From left to right, the lengths of the boxes are $\ell_1, \ldots, \ell_n$. You want to tie up the boxes using strings. You have an unlimited supply of strings, but one string can only be used to tie up a *contiguous block* of boxes with total length is at most L. You want to compute an optimal solution that uses fewest possible strings to tie up all the boxes.

Part (a) [3 marks]

What is the necessary and sufficient condition for it to be possible to tie up all the boxes (using as many strings as needed)? No justification is needed.

Part (b) [2 marks]

What is the minimum and maximum number of strings that might ever be needed? No justification is needed.

Part (c) [5 marks]

Design an $O(n)$ time greedy algorithm for computing a solution that uses the fewest possible strings. Argue why your algorithm runs in $O(n)$ time.

Part (d) [5 marks]

Prove that your greedy algorithm always returns an optimal solution.

Question 2. Dynamic Programming [20 marks]

There are n dice of different colours. These are regular dice with numbers $1, \ldots, 6$ printed on their six sides. Given an integer m, you want to calculate the number of ways in which you can roll the dice to get the sum of numbers to be m. For example, with $n = 2$ dice, there are three ways of getting the sum $m = 3$: the two dice can roll $(1, 2)$ or $(2, 1)$. Note that the dice have different colours, so they are unique.

Part (a) [10 marks]

Write the Bellman equation for dice(n, m), which is the number of ways of getting sum m out of n dice. Clearly identify your base cases. Briefly justify why your Bellman equation is correct.

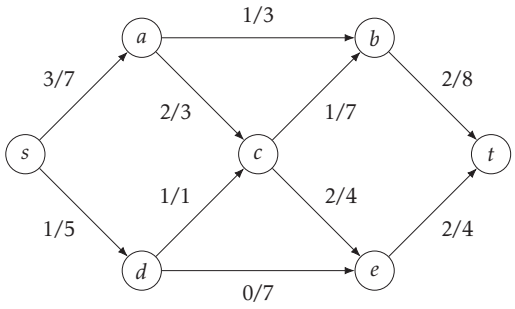
Part (b) [5 marks]

Write a bottom-up dynamic program that implements your Bellman equation and computes dice(n, m) in $O(n \cdot m)$ time.

Part (c) [5 marks]
How would you modify your algorithm from part (b) so that it runs in time $O(n \cdot \min(n, m))$?

Question 3. Network Flow I [15 marks]

Consider the following network N and an initial flow f. Each edge is labeled a/b, where b is the capacity of the edge and a is the flow it carries under f.



Part (a) [6 marks]

Find a maximum s-t flow in this network N by starting from f, and using augmenting paths in the residual graph as in the Ford-Fulkerson algorithm. Show your work. For each iteration, state the augmenting path you use (including which edges are forward and which are reverse), the amount of additional flow you send along the path, and the total value of the flow at the end of the iteration.

Part (b) [3 marks]

Based on your maximum flow from (a), find a minimum s-t cut in network N. List the edges that go across your cut from the s side to the t side. No other justification is needed.

Part (c) [3 marks]

From the edges listed in part (b) that go across the min s-t cut, find an edge such that increasing its capacity by one would increase the maximum s-t flow. State the edge and find an augmenting path in the residual graph of your flow from part (a) after increasing the capacity of this edge by one.

Part (d) [3 marks]

Give an example of one edge across your cut in part (b) such that increasing its capacity by one would NOT increase the max flow. State the edge, and explain why this is possible despite the max-flow min-cut theorem which states that the max s-t flow in a network is equal to the minimum capacity of any s-t cut.

Question 4. Network Flow II [10 marks]

There are n families who go to a resort together for a vacation. Family i consists of x_i people. There are m activities available at the resort, and at most y_j people are allowed to participate in activity j. As the trip planner, your job is to assign people from all families to activities. But you need to make sure that each person from each family is assigned to exactly one activity, no activity j is assigned more than y_j people, and no two people from the same family are assigned to the same activity.

By reducing this problem to network flow and assuming you are given access to a polynomial-time algorithm for network flow, design a polynomial-time algorithm that either produces a feasible assignment of people to activities or declares that no such assignment exists.

Part (a) [2 marks]

TRUE/FALSE: "*An algorithm is polynomial-time if it runs in time polynomial in n, m, $x_1, \ldots, x_n$, and $y_1, \ldots, y_m$.*" (If this is true, explain why. If this is false, describe the quantities in which the running time should really be polynomial.)

Part (b) [8 marks]

Describe your flow network. Clearly state the vertices (including source and target vertices), the edges, and the edge capacities. Explain how computing the maximum flow in this network helps you solve your activity planning problem.

Question 5. Linear Programming [15 marks]

Part (a) [5 marks]

A cargo plane has two sections (front and rear), whose weight and volume limits are given in the first table below. These sections can be loaded with two types of cargo, whose volume and profit per unit weight are given in the second table below.

Section	Weight limit (tonnes)	Volume limit (m^3)
Front	5	300
Rear	3	200

Cargo	Volume (m^3/tonne)	Profit (CA$/tonne)
C1	35	15
C2	53	32

Write a linear program that decides how many tonnes of each cargo should be loaded into each section to maximize profit without violating the weight and volume limits of each section. Clearly indicate what each variable of your LP means. Assume that you have unlimited supply of each cargo and that the number of tonnes loaded can be any non-negative real number. (You only need to write the LP; you do not need to actually solve it.)

December 2019 Examinations

CSC 373 H1F

Duration: **3 hours**

Part (b) [5 marks]

Consider the following program, which is not linear because one of its constraints involves absolute values. Show that this can be converted to an equivalent linear program by replacing the problematic constraint by one or more linear constraints. Argue why your LP is equivalent (i.e. produces the same optimal solution). Here, $|\cdot|$ is the absolute value function, i.e., $|z| = z$ if $z \geqslant 0$ and $|z| = -z$ if $z < 0$.

Maximize $2a - 3b$
s.t.
$|a-5|+|a+b-3| \leqslant 10$
$a, b \geqslant 0$

Part (c) [5 marks]

Consider the following program, which is not linear because the objective function involves absolute value. Show that it can still be solved in polynomial time by solving one or more LPs.

Maximize $|c^T x|$
s.t.
$Ax \leqslant b$
$x \geqslant 0$

Question 6. Complexity [10 marks]

Part (a) [5 marks]

Consider the following two problems.

HAM-CYCLE:
Input: An undirected graph $G = (V, E)$.
Question: Does G have a Hamiltonian cycle (i.e. a simple cycle containing all $|V|$ vertices)?

HALF-CYCLE:
Input: An undirected graph $G = (V, E)$.
Question: Does G have a simple cycle containing at least $\lfloor\, |V|/2 \,\rfloor$ vertices?

Show that HALF-CYCLE is NP-complete. For the hardness part, use HAM-CYCLE in your reduction. (You can assume that HAM-CYCLE is known to be NP-complete.)

Part (b) [5 marks]

Consider the following two problems. They are decision versions of the corresponding optimization problems that we saw in class.

MAX-CUT:
Input: An undirected graph $G = (V, E)$ and an integer ℓ.
Question: Is there a partition of V into (A, B) such that at least ℓ edges have one endpoint in A and the other in B?

MAX-2-SAT:
Input: A 2-CNF formula φ and an integer t. (In a 2-CNF formula, each clause has exactly two literals.)
Question: Does there exist a truth assignment under which at least t clauses of φ are satisfied?

Show that MAX-2-SAT is NP-complete. For the hardness part, use MAX-CUT in your reduction. (You can assume that MAX-CUT is known to be NP-complete.)

[Hint: You can construct a 2-CNF formula which has two clauses for every edge in G.]

Question 7. Approximation Algorithms [15 marks]

Consider the following problem.

d-REGULAR-MAX-INDEPENDENT-SET:
Input: An undirected graph $G = (V, E)$ in which every vertex has degree exactly d.
Output: A maximum cardinality independent set I^* of G (an independent set is a subset of vertices such that no two of them are connected by an edge).

Consider the following greedy algorithm for this problem.

```
GreedyIndSet(G):
    I ← ∅;
    while G has at least one remaining vertex:
        v ← any remaining vertex in G;
        I ← I ∪ {v};
        Delete v and all its neighbours from G;
    return I
```

Part (a) [5 marks]

Argue that the greedy algorithm always returns an independent set.

Part (b) [10 marks]

Find the largest c such that the greedy algorithm is guaranteed to return an independent set containing at least $c \cdot |V|$ vertices. Your answer should be in terms of only d. Justify your answer.

(Bonus) Part (c) [10 marks]

Find the largest ρ such that the greedy algorithm is a ρ-approximation. That is, on any graph G, it produces an independent set of cardinality at least $\rho \cdot OPT$, where OPT is the maximum cardinality of any independent set in G. Your answer should be in terms of only d. Justify your answer.

NOTE: This subquestion is for extra credit. Solve this only after attempting all other questions.

December 2019 Examinations

CSC 373 H1F

Duration: **3 hours**

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Total Marks = 100