PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

APRIL 2016 EXAMINATIONS

CSC 373 H1S
Instructor: Allan Jepson

Duration — 3 hours

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

Student Number:	
Family Name(s):	
Given Name(s):	

---

*Do **not** turn this page until you have received the signal to start.*
In the meantime, please fill in the identification section above and read the instructions below carefully.

This final examination paper consists of 6 questions on 14 pages (including this one), printed on one side of each sheet.

Answer each question directly on this paper, in the space provided. For rough work you can use the reverse side of the pages or the blank last page. If you need more space for one of your solutions, use the reverse side of a page or the last 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, or tests, 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.

I am asked to remind you that you must obtain no less than 35% on this exam in order to pass this course.

MARKING GUIDE

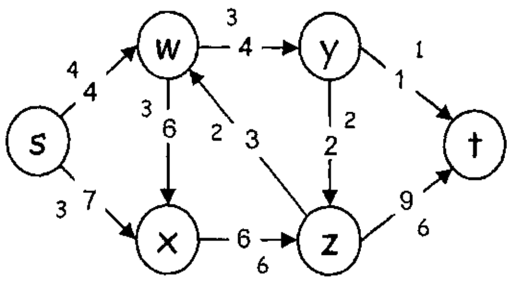
# 1:	______/ 15
# 2:	______/ 20
# 3:	______/ 20
# 4:	______/ 15
# 5:	______/ 15
# 6:	______/ 15
TOTAL:	______/100

Good Luck!

CONT'D...

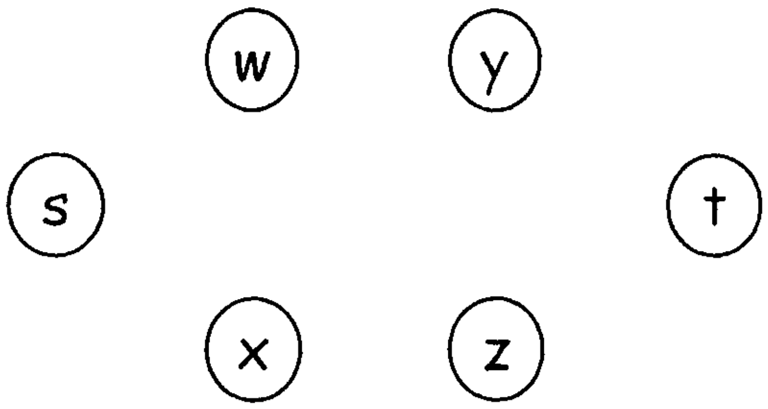
Question 1. Ford-Fulkerson Algorithm [15 MARKS]

Let $G = (V, E, c)$ be the s-t network flow problem drawn below. For each edge $e \in E$ we display the capacity $c(e)$ as an integer placed on the edge e, while the flow $f(e)$ is given by the number in smaller font nearby.



Part (a) [5 MARKS]

Draw the residual graph G_f for this flow (for your convenience the vertices in the above graph have been copied below).

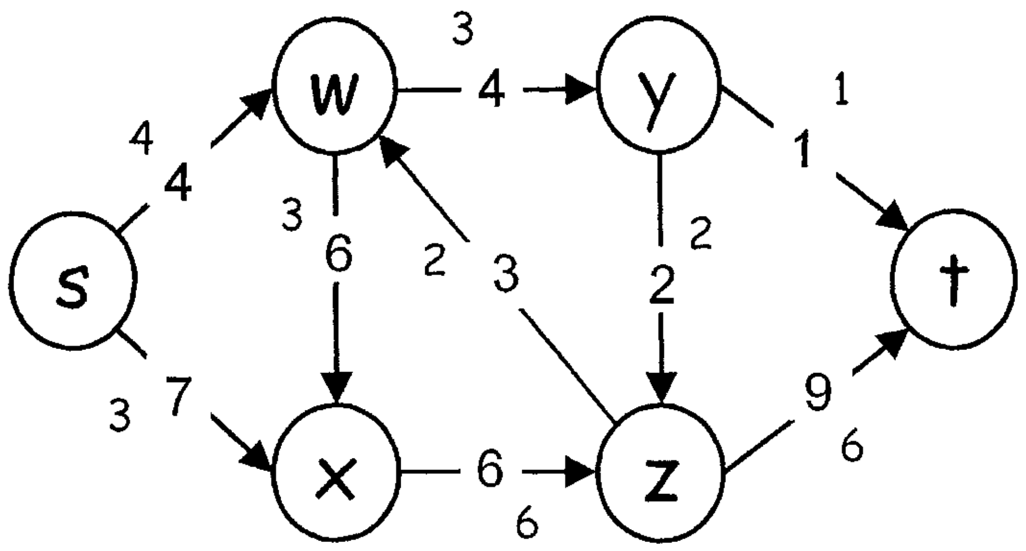


Part (b) [2 MARKS]

Indicate an augmenting path on the residual graph G_f (say, by listing the vertices on that path in order). What is the maximum amount of flow that can be pushed along this augmenting path?

Part (c) [3 MARKS]

The original network flow from part (a) is redrawn again below. Suppose one step of the Ford-Fulkerson algorithm is performed on G_f using the augmenting path from part (b). Indicate the resulting change to the flow in the drawing below (by crossing out flow values that need to be updated and writing in the updated values).



Part (d) [5 MARKS]

Show that the resulting flow is a maximum flow on G by exhibiting an appropriate s-t cut on G. Explain why this cut shows the flow is maximal.

Question 2. P & NP Definitions [20 MARKS]

In the lectures we defined the sets P, NP, co-NP, NP-complete, and NP-hard.

For each problem below, provide a list of **ALL** of the above sets that the problem is either known to be a member of, or likely to be a member of. For each of the above sets that you include in your answer (and only for those ones) give a brief reason for including it, possibly by either quoting results in the lecture notes or tutorial exercises, or by stating a poly-time certifier, disqualifier, and/or reduction that would be convenient to use to prove part of your answer. No proofs are required.

By poly-time or poly-size we mean bounded by a polynomial in $|s|$, the size of the input. For each problem below we provide a suitable choice for $|s|$ (actually a simple lower bound of the input size, but sufficient for our purposes). We define bits(k) to be the number of bits needed to represent the integer k, e.g., you can take bits(k) $= \text{ceil}(\log_2(k)) + 2$.

Part (a) [5 MARKS]

Input: x, where x is postive integer.
Input Size: Use $|s| = \text{bits}(x)$.
Problem A: Is x a prime number?

Part (b) [5 MARKS]

Input: (U, F, k), where

- U is a finite set,
- $F = \{F_1, \ldots, F_M\}$ is a set of subsets $F_m \subseteq U$, for $1 \le m \le M$,
- k is a postive integer.

Input Size: Use $|s| = |U| + |F| + \text{bits}(k)$.
Problem B: Does there exist a subset $C \subseteq \{1, 2, \ldots, M\}$ such that $|C| \ge k$ and, for each $i, j \in C$ with $i \neq j$, we have $F_i \cap F_j = \varnothing$?

Question 2. (CONTINUED)

Part (c) [5 MARKS]

Notation: We use $\bigwedge_{m=1}^{M} C_m(X)$ to denote the conjunction $C_1(X) \wedge \ldots \wedge C_M(X)$, and use $\bigwedge_{m \in S} C_m$ to denote the conjunctions of all clauses C_m with $m \in S$.
Input: $(\Phi, \{w_m\}_{m=1}^{M}, W)$, where

- $\Phi(X) = \bigwedge_{m=1}^{M} C_m(X)$ is a CNF formula with disjunctive clauses $C_m(X)$ for $m = 1, 2, \ldots, M$. Here $X = (x_1, x_2, \ldots, x_N)$ denotes the tuple of logical variables involved in Φ. Each disjunctive clause $C_m(X)$ has at most L literals (see Question 5), and we assume $L \geq 3$.
- $\{w_m\}_{m=1}^{M}$ is a set of integer-valued weights, one for each clause C_m in Φ.
- W is an integer.

Input Size: Use $|s| = L + M + b_{max}$, where b_{max} is the maximum number of bits needed to represent W or any weight w_m.
Problem C: Does there exist a subset $S \subseteq \{1, 2, \ldots, M\}$ such that $\bigwedge_{m \in S} C_m(X)$ is satisfiable and $\sum_{m \in S} w_m \geq W$?

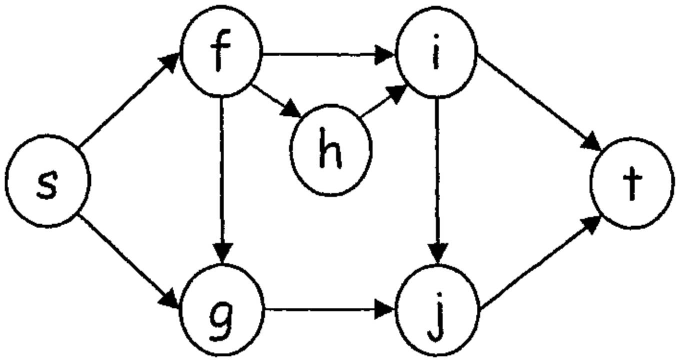
Part (d) [5 MARKS]
Input: (Φ, k) where

- $\Phi(X) = \bigwedge_{m=1}^{M} C_m(X)$ is as in part (c) above,
- k is a postive integer.

Input Size: Use $|s| = L + M + \text{bits}(k)$, where L and M are as in part (c) above.
Problem D: Is $\bigwedge_{m \in S} C_m(X)$ unsatisfiable for all $S \subseteq \{1, 2, \ldots, M\}$ with $|S| = k$?

Question 3. Dynamic Programming [20 MARKS]

We are given a directed acyclic graph $G = (V, E)$ with two distinguished vertices s and t such that: a) the vertex t is reachable from all $v \in V$; and b) every vertex $v \in V$ is reachable from s (see the example below). (By reachability we mean there must be a path which traverses edges in the prescribed direction. For example, in the figure below, only t is reachable from t itself.)



For each vertex $v \in V$, define $p(v)$ to be the set of parent vertices of v, that is, $p(v) = \{u \in V \mid (u, v) \in E\}$. Similarly, let $c(v)$ denote the child vertices of v, that is, $c(v) = \{u \in V \mid (v, u) \in E\}$. For example, in the graph drawn above, the parents of vertex $i \in V$ are $p(i) = \{f, h\}$ and the children are $c(i) = \{j, t\}$.

We wish to count the number of distinct paths from s to t in G using a specific dynamic programming approach. We define two s-t paths P and Q in G to be distinct if there is some edge e in P that is not in Q, or vice versa. For example, the paths $P = (s, g, j, t)$ and $Q = (s, f, g, j, t)$ are considered to be distinct because there is at least one edge in Q (e.g., (f, g)) that is not in P.

Suppose $N(v)$ denotes the number of distinct paths from s to v in G. We define $N(s) = 1$ to denote the path of length zero.

Part (a) [3 MARKS]

In the figure above, write the integer value of $N(v)$ beside each vertex in the graph. For example, you should write 1 beside vertex s and 1 beside vertex f. How many distinct paths are there from s to t?

Part (b) [5 MARKS]

For a general input graph G, as described above, show how to determine $N(v)$, for any $v \in V \backslash \{s\}$, in terms of the values of $N(u)$ at each $u \in p(v)$. Explain why your expression is correct.

Question 3. (CONTINUED)

Part (c) [12 MARKS]

Write a pseudo-code algorithm to compute $N(v)$ for all $v \in V$. To get any marks, the algorithm **must not make any use of recursion**. Moreover, it should run in $O(|V| + |E|)$ time, and the only non-trivial data structures or functions that it can use are:

- the set of vertices V, including the start and end vertices s and t;
- the functions $c(v)$ and $p(v)$ described above, which effectively define the edges in the graph;
- a read/write dictionary (or array) $N(v)$, where $v \in V$ and $N(v)$ can store a non-negative integer;
- a queue data structure which can maintain a queue of vertices;
- iterators over sets (i.e., you can loop over all elements in a set).

In addition to the pseudo-code, briefly explain why your algorithm is correct, and why it runs in $O(|V|+|E|)$ time.

Question 4. s-t Min Cuts [15 MARKS]

Consider a s-t network flow problem $G = (V, E, c)$, where $c(e)$ denotes the edge capacity for each $e \in E$, and $c(e)$ is a positive integer.

Suppose we are given an integer-valued max flow f for this problem. Let G_f be the residual graph and (A_1, B_1) the min s-t cut, where both G_f and (A_1, B_1) are as defined by the algorithms described in the lecture notes. In addition, define

$$B_2 = \{u \in V \mid t \text{ is reachable from } u \text{ in the residual graph } G_f \},$$
$$A_2 = V \backslash B_2.$$

Note that $t \in B_2$ since t is always reachable from itself.

Part (a) [2 MARKS]

Show an example of an s-t network flow problem, and a maximum flow, where the two s-t cuts defined above are different, that is, $A_1 \neq A_2$. For your example, explain which vertices are in A_1 and which are in A_2.

Part (b) [5 MARKS]

For a general s-t network flow problem (i.e., as described in the beginning of this question and not just your example in part (a)), suppose f is an integer-valued max flow. Prove the s-t cut (A_2, B_2) defined above is a minimum capacity s-t cut.

Question 4. (CONTINUED)

Part (c) [3 MARKS]

Following on from part (b), show that $A_1 \subseteq A_2$ and $B_2 \subseteq B_1$. (Hint: Consider the statement "$a \in A_1$ implies $a \notin B_2$".)

Part (d) [5 MARKS]

We are interested in identifying a **critical edge** in G. That is, suppose we consider an modified nework flow problem $G' = (V, E, c')$ which is the same as the original problem G except the capacity of one edge, namely $e_0 = (u_0, v_0) \in E$, has been increased by one (i.e., $c'(e_0) = c(e_0) + 1$ but otherwise $c'(e) = c(e)$ for all $e \in E \backslash \{e_0\}$). We say the edge $e_0 = (u_0, v_0) \in E$ is critical with respect to the max flow f of G if and only if the max flow of G' has value $v(f) + 1$. That is, increasing the capacity of a critical edge e_0 by one allows the value of max flow to increase by one.

Show that $e_0 = (u_0, v_0)$ is a critical edge with respect to f iff $u_0 \in A_1$ and $v_0 \in B_2$. You can use the result in part (c) even if you did not provide an answer for that part.

Question 5. Integer Linear Programming [15 MARKS]

Consider a 3-SAT formula $C_1 \wedge C_2 \wedge \ldots \wedge C_M$ which involves the boolean (logical) variables $X = (x_1, x_2, \ldots, x_N)$. Each clause C_m is a disjunction of three literals involving three distinct variables x_j. (A literal is either x_j of its logical negation $\bar{x_j}$.) We write $C_m(X)$ to denote the dependence of C_m on X.

Suppose we are also given non-negative weights w_m for each clause C_m. Given an X, define $S(X)$ to be the indicies of the clauses that are true for X, that is $S(X) = \{m \mid C_m(X) \text{ is true and } 1 \leq m \leq M\}$. Define the profit for X to be

$$P(X) = \sum_{k \in S(X)} w_k \tag{1}$$

That is, each clause $C_k(X)$ contributes the weight w_k to the above sum when that clause is true and it contributes zero otherwise.

Here we consider a poly-time reduction of this weighted 3-SAT problem to an integer linear programming problem (ILP). The reduction must use an encoding where y_n is a variable in your ILP, with $y_n \in \{0, 1\}$, and

$$x_k \text{ is true if and only if } y_k = 1. \tag{2}$$

You can include other integer-valued variables in your ILP and, if you do, make sure you describe what purpose these additional variables serve.

Part (a) [5 MARKS]

As a warm-up exercise, suppose the first clause is $C_1 = (x_5 \vee x_2 \vee \bar{x_3})$. Specify a set of linear constraints on the variables y_2, y_5, and y_3 that are all satisfied iff the clause C_1 is true (use the relationship between x_k and y_k given in equation (2)).

Part (b) [5 MARKS]

As a second warm-up exercise, consider the simple case in which there is only one clause in Φ, so $M = 1$. In this case specify an ILP which has a maximum profit of w_1 whenever the variables y_k, for $1 \leq k \leq N$, imply that $C_1(X)$ must be true (i.e., where each x_k is determined from y_k and eqn (2)), and the ILP has a maximum profit of zero otherwise.

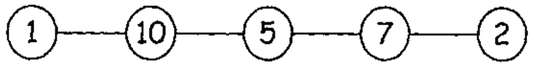
Part (c) [5 MARKS]

Show how to reduce the original weighted 3-SAT problem to an ILP, where the relationship between the ILP variables y_k and the logical variables x_k are as in equation (2). You should clearly describe the ILP, and explain why the maximum profit of your ILP equals the maximum profit of the weighted 3-SAT problem. (You do not need to provide proofs.)

Question 6. Approximation Algorithms [15 MARKS]

Recall that an independent set S of a graph $G = (V, E)$ is defined to be a subset $S \subseteq V$ such that, for each edge $e \in E$, at most one endpoint of e can be in S. We consider a weighted version of this problem where each vertex $v \in V$ is associated with a positive weight $w(v)$, and we seek an independent set S with the maximum weight. (The weight of S is defined to be $W(S) = \sum_{v \in S} w(v)$.)

Moreover, we consider this problem for the simple case in which $G = (V, E)$ is just a path (i.e., a spanning tree of degree at most 2). For example, G could be



where the weight of each vertex is the integer displayed inside it.

Consider the following greedy algorithm for this weighted independent set problem:

```
[S] = wIndSet(V, E, w)
Initialize $F \leftarrow (V, E)$ and $S \leftarrow \{\ \}$.
While the graph $F$ is not empty:
    Find a vertex $u$ in $F$ with the largest weight $w(u)$.
    $S \leftarrow S \cup \{u\}$
    Update $F$ by deleting the vertex $u$ and all its neighbouring vertices $v$ (i.e., all vertices $v$ with
    an edge $(u, v)$ still in $F$), and delete all the edges ending at any of these deleted vertices.
End while
return S
```

For the above example, this algorithm selects the vertex with weight 10 first, removes it along with its two neighbours (with weights 1 and 5), and also removes the three leftmost edges in the above figure. Next it selects the vertex with weight 7, and deletes both remaining vertices and the remaining edge. It returns S with weight $w(S) = 17$, which is the maximum weight for any independent set for this example.

Part (a) [2 MARKS]

Show that, when $G = (V, E)$ is a path, this greedy algorithm does not always find an independent set of maximum weight.

Part (b) [5 MARKS]

Given any small real value $\epsilon > 0$, show an example of a weighted path G for which the above greedy algorithm returns an independent set S with an approximation ratio of $(2 - \epsilon)$. That is, suppose the weight of the returned set is $w(S)$ and the maximum possible weight is $w(S^*)$, then your example needs to have $(2 - \epsilon)w(S) = w(S^*)$. (Note G must be a path and all the weights $w(v)$ must be positive. You can use real-valued weights for this example.) Briefly explain what S and S^* are for your example, along with the ratio $w(S^*)/w(S)$ of their weights.

Question 6. (CONTINUED)

Part (c) [8 MARKS]

Prove that, when $G = (V, E)$ is a path, the above algorithm is a 2-approximation for finding an independent set of maximum weight.

For Scratch Work

Total Marks = 100