CS 4102 Unit D Exam

Name Winston Zhang

You MUST write your e-mail ID on **EACH** page, and put your name on the top of this page, too.

You MUST write each answer on the same page as the question in the space provided for an answer. (This is necessary because we scan exams and grade them using GradeScope.)

If you are still writing when "pens down" is called, your exam will be ripped up and not graded.

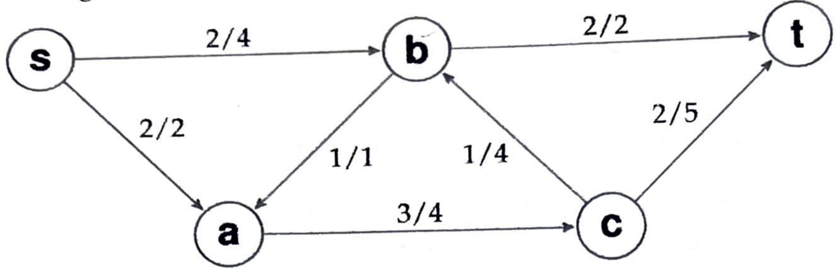
There are 6 pages to this exam, including this page. There are a total of 100 points on the exam. Once the exam starts, please make sure you have all the pages. Questions are worth different amounts, so be sure to look over all the questions and plan your time accordingly.

This exam is CLOSED text book, closed-notes, closed-cell phone, closed-smart watch, closed-computer, closed-neighbor, etc. It is an honor violation to communicate information about the exam to anyone who might not have yet taken it. Please write and sign the honor pledge below.

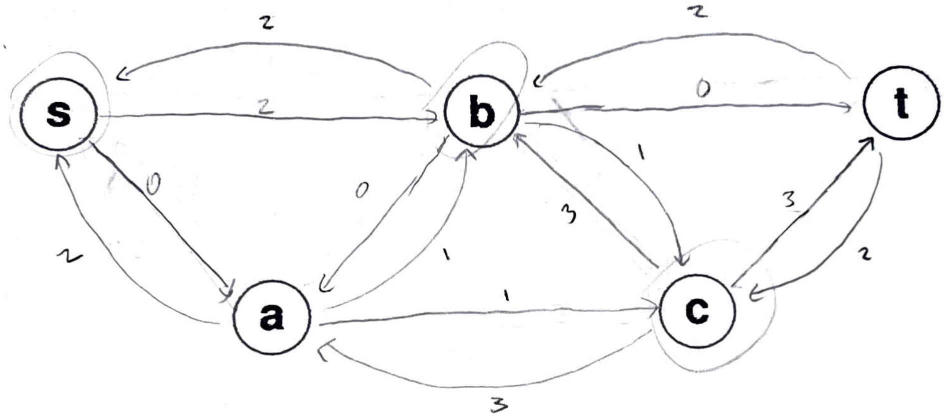
On my honor as a student, I have neither given nor received unauthorized aid on this exam.

Page 2: Max Flow

Given the following Flow Network G:



1. [14 points] Complete the Residual Graph G_f below by drawing in and labeling the edges.



2. [5 points] Find an augmenting path in the graph G_f. List the nodes in the path you found in order (e.g., $s \to a \to b \to c \to t$).

s→b→c→t

3. [5 points] Find the min cut of the graph. List the nodes below on each side of the cut.

S (one side)	$V - S$ (other side)
s, b	a, c, t

4. [4 points] What is the maximum flow of this graph? 5

Page 3: Reductions

5. [15 points] Reduce *Element Uniqueness* to *Mode* in $O(n)$ time.
The problem *Element Uniqueness* is defined as: given a list of numbers $\{a_1, a_2, ..., a_n\}$, return true if no number appears more than once (i.e., every number is distinct).
The problem *Mode* is defined as: given a list of numbers $\{a_1, a_2, ..., a_n\}$, return one of the numbers which appears most frequently in the list. If all the numbers occur the same number of times, it will return an arbitrary element.

Refer to $\{a_1, a_2, \ldots, a_n\}$ as A.

Run Mode(A), storing the returned integer as a key to search for in A. Find the first instance of Mode(A) in A, and remove it ($\Theta(n)$), and then search the list for the same integer value. If it is found again, Element Uniqueness returns false, if it is not found, return true.

The algorithm that uses the solution to Mode() to solve the element uniqueness problem has a linear running time worst case.

In class we taught you about bipartite matching using an example where TAs were matched with dogs they can adopt. Here are questions about a variant of this problem and how we reduce it to G', an instance of network flow. The changes to the problem are:

- R, the set of dogs, is larger than L, the set of TAs.
- We will allow at most two TAs to have "joint custody" of one dog, meaning they both adopt a dog together. So a dog can be adopted by one or two TAs.
- Each TA can only adopt one dog.

Given these changes, answer the following 4 questions:

6. [3 points] In G', the network flow graph, what capacity should be assigned to each edge connecting s (the source) to a node in L (the set of TAs)? $\boxed{1}$
7. [3 points] What capacity should be assigned to each edge connecting a node in L (the set of TAs) to a node in R (the set of dogs)? $\boxed{1}$
8. [3 points] What capacity should be assigned to each edge connecting a node in R (the set of dogs) to t (the sink)? $\boxed{2}$
9. [3 points] Fill in the circle *completely* if you think the following statement is **true**.
   ○ Every dog has been matched with a TA if the maxflow value on G' equals $|R|$.

Page 4: Reductions and NP-Completeness

10. [8 points] To show that a problem A is NP-complete, we must show that $A \in$ NP and $A \in$ NP-hard. What must we show and prove for each case? (Be as specific and brief as possible).

a) $A \in$ NP   Show that A can be solved in polynomial time non-deterministically, by constructing a non deterministic finite state machine;
Or show that a proposed solution to A can be verified in polynomial time

b) $A \in$ NP-hard   Show that the problem is at least as hard as any other NP hard problem.
Show that a known NP-Hard problem reduces to A in polynomial time.

11. [5 points] In no more than one sentence, state the Cook-Levin Theorem.

The Satisfiability problem is a member of the set of NP-Complete problems.

12. [8 points] Consider the reduction to show that *k-Independent-Set* (k-IndSet) is NP-Hard. For each of the following statements, fill in the circle completely if the statement is **true**. *No justification needed.*

- ○ To show that *k-IndSet* is NP-Hard, we reduced it to the 3-CNF (i.e., 3-SAT) problem.
- ● The three literals/variables in each clause of the 3-CNF expression become a 3-clique in the graph for the k-IndSet instance.
- ○ All pairs of vertices in the graph for the k-IndSet instance are connected to each other *except* where a pair corresponds to a logical variable and its negation in the 3-CNF expression.
- ○ There is a k-independent set in the graph created by the reduction if and only if there is a satisfying assignment for the logical expression, where k is the total number of literals in the expression.

Page 5: More on NP Completeness

13. [10 points] Let A and B be arbitrary problems. For each of the following statements, fill in the circle completely if the statement is **true**. *No justification needed.* **Note:** $\varnothing$ is the empty set.

    - ● If $A, B \in$ NP-complete, then $A \leq_p B$.
    - ● If there exists a problem A where $A \in$ P $\cap$ NP-hard, then P $=$ NP.
    - ○ If $A \notin$ P, then $A \in$ NP.
    - ○ If P $=$ NP, then P $\cap$ NP-complete $= \varnothing$.
    - ● If P $=$ NP and $A \in$ NP-complete, then there is a polynomial-time algorithm for A.

14. [4 points] (Answer the following with True, False or Unknown.)
    NP-Complete problems like *k-Vertex Cover* and 3-CNF (i.e., *3-SAT*) cannot be solved in less than exponential time.

    Unknown

15. [10 points] Which of the following are part of what you would need to do show that the *k-Vertex Cover* problem belongs to NP, if you were given the inputs G, k, plus a certificate S that could represent set of vertices? Fill in the circle completely for each of the items below that *are* a part of what you need to do.

    - ● Check that $|S| = k$.
    - ○ Check that an edge exists in G that connects each vertex in S to at least one other vertex in S.
    - ● Check if each edge in G is incident to *at least* one vertex in S.
    - ○ Check if each edge in G is incident to *at most* one vertex in S.
    - ● Make sure that each of the checks you do takes no more time than than $O(V^c)$, where c is some constant.