NAME: ____________________________

CMPSCI 611
Advanced Algorithms
Final Exam Fall 2015

A. McGregor

17 December 2015

DIRECTIONS:

- Do not turn over the page until you are told to do so.
- This is a *closed book exam.* No communicating with other students, or looking at notes, or using electronic devices. You may ask the professor to clarify the meaning of a question but do so in a way that causes minimal disruption.
- If you finish early, you may leave early but do so as quietly as possible. The exam script should be given to the professor.
- There are five questions. Some questions may be easier than others. Don't spend too long on a problem if you're stuck – you may find that there are other easier questions.
- The front and back of the pages can be used for solutions. There is also a blank page at the end that can be used. If you are using these pages, clearly indicate which question you're answering.
- The exam will finish at 15:00 pm.
- Good luck!

1	/10
2	/10
3	/10
4	/10
5	/8+3
Total	/48+3

Question 1 (True or False): For each of the following statements, indicate whether the statement is true or false by circling the appropriate option. It is not necessary to give justification.

1. Given a bipartite graph, the maximum matching can be found in polynomial time.

   - TRUE
   - FALSE

2. The max value of $2x_1 + x_2$ subject to the constraints $x_1 \geq 0, x_2 \geq 0, x_1 + x_2 \leq 1$ is 2.

   - TRUE
   - FALSE

3. The Simplex algorithm runs in polynomial time.

   - TRUE
   - FALSE

4. If every capacity in a flow network is integral then the maximum flow between any two nodes is integral.

   - TRUE
   - FALSE

5. For any random variable X, $\mathbb{P}\left[(X - \mathbb{E}[X])^2 \geq t^2\right] \leq \mathbb{V}[X]/t^2$.

   - TRUE
   - FALSE

Question 2 (Numbers): Let $A = \{a_1, a_2, \ldots, a_n\}$ be a set of positive integers where n is odd. You may assume that the integers are distinct. The elements are not sorted.

1. Assuming any operation on two numbers in A takes $O(1)$ time, how fast can each of the following operations be performed (no justification required):

   (a) Sorting the elements of A:

   (b) Computing $\max(A)$:

   (c) Computing $\text{median}(A)$, i.e., the element with rank $(n+1)/2$:

   (d) Computing $\sum_{1 \le i < n} (a_{i+1} - a_i)$:

   (e) Computing $\sum_{1 \le i < j \le n} (a_i - a_j)^2$:

2. Given the set A and a positive integer t, consider the problem of determining whether there is a subset of A that sums to t. Each value of A may be used at most once and the values in the empty subset sum to 0.

   (a) Let M be an $n \times t$ table where $M[i,j] = 1$ if there is a subset of $\{a_1, \ldots, a_i\}$ that sums to j and $M[i,j] = 0$ otherwise. Give a formula relating $M[i,j]$ to other entries of the table and justify your answer.

   $$M[i,j] =$$

   (b) Write an algorithm for determining whether there is a subset of A that sums to t. What is the running time of your algorithm in terms of n and t?

Question 3 (Reduction and Formulas):

1. Briefly explain why, if you design a polynomial time algorithm for any NP complete problem, then you have proved $P = NP$.

An instance of 3-SAT has exactly 3 literals per clause and we want to determine if there is a setting of the variables such that the formula is satisfied. A related problem is k-SAT where every clause has exactly k literals. For example, $(x_1 \vee \bar{x}_2) \wedge (x_3 \vee x_4)$ is an instance of 2-SAT. Using the fact that 3-SAT is NP-complete and 2-SAT is in P, indicate if each of the following two statements are TRUE or FALSE and *give a brief justification.*

2. If $P \neq NP$ then 3-SAT $\leq_P$ 2-SAT, i.e., any 3-SAT instance ϕ can be transformed (in polynomial time) into a 2-SAT instance that is satisfiable iff ϕ is satisfiable.

3. If $P \neq NP$ then 2-SAT $\leq_P$ 3-SAT.

We next consider a polynomial time reduction from k-SAT to 3-SAT for any $k > 3$. Given an instance ϕ of k-SAT we create an instance 3-SAT as follows: for each clause $C = (a_1 \vee a_2 \vee \ldots \vee a_k)$ (where a_i are literals) of ϕ, we introduce $k-3$ new variables $y_1, \ldots, y_{k-3}$ and replace C by

$$f(C) = (a_1 \vee a_2 \vee y_1) \wedge (\bar{y}_1 \vee a_3 \vee y_2) \wedge \ldots \wedge (\bar{y}_{k-4} \vee a_{k-2} \vee y_{k-3}) \wedge (\bar{y}_{k-3} \vee a_{k-1} \vee a_k) \ .$$

4. Show that if C is satisfied there is a setting of $y_1, \ldots, y_{k-3}$ such that $f(C)$ is satisfied.

5. Show that if $f(C)$ is satisfied then C is also satisfied.

Question 4 (Assigning Tasks): Suppose there are n tasks to be done and $m < n$ people who are available to do these tasks. Task i takes t_i minutes to complete and assume that $t_1 \geq t_2 \geq \ldots \geq t_n > 0$. Your goal is to assign the tasks to the m people such that everyone finishes in the smallest amount of time. Let $\textsc{opt}$ be the smallest value such that it is possible to assign the tasks such that everyone finishes in $\leq \textsc{opt}$ time. For example, if $n = 4$, $m = 3$, $t_1 = 4, t_2 = 3, t_3 = 2$, and $t_4 = 2$ then $\textsc{opt} = 4$ since we can assign the first task to the first person, the second task to the second person, and the remaining tasks to the third person.

1. For arbitrary $t_1, \ldots, t_n$, explain why $\textsc{opt} \geq t_1$?

2. For arbitrary $t_1, \ldots, t_n$, explain why $\textsc{opt} \geq \sum_{i=1}^{n} t_i/m$?

3. For arbitrary $t_1, \ldots, t_n$, explain why $\textsc{opt} \geq t_m + t_{m+1}$?

Consider the algorithm which assigns the tasks in order (i.e., the longest task is assigned first) and gives task i to the person who has currently been assigned the least number of minutes.

1. After we have assigned the first m tasks, show that everyone has been assigned at most $\textsc{opt}$ minutes of work.

2. After we have assigned the first n tasks, show that everyone has been assigned at most $1.5 \times \textsc{opt}$ minutes of work. Hence, the algorithm is a 1.5 approximation.

Question 5 (Randomized Algorithm for Independent Set): Let G be an undirected graph with n nodes and m edges where $n \leq 2m$. Recall that an independent set of G is a subset U of the vertices such that there does not exist an edge whose endpoints are both in U. Consider the following randomized algorithm for finding an independent set:

Step 1. Delete each node (and its incident edges) with probability $1 - p$ where $p = \frac{n}{2m}$.

Step 2. If there is an edge remaining, delete one of the endpoints (and its incident edges). Repeat until there are no edges remaining.

Let U be the set of nodes that have not been deleted.

1. Argue that U is an independent set.

2. Let X be the number of nodes that remain after Step 1. What is the value of $\mathbb{E}[X]$? Justify your answer.

3. Let Y be the number of edges that remain after Step 1. What is the value of $\mathbb{E}[Y]$? Justify your answer.

4. Prove that the expected size of U is at least $\frac{n^2}{4m}$. *Hint:* Bound $|U|$ in terms of X and Y.

5. *Extra Credit:* Showing that the expected size of U is at least $\frac{n^2}{4m}$, implies that every graph with n nodes and m edges has an independent set of size at least $\frac{n^2}{4m}$. Why? Can you use similar reasoning to show that if

$$\binom{n}{r} < 2^{\binom{r}{2}-1}$$

then there exists a way to color the $\binom{n}{2}$ edges of a complete graph on n nodes with just two colors such that there is no clique of size r where every edge in the clique has the same color. *Hint:* Consider coloring the edges randomly and compute the expected number of cliques of size r where every edge has the same color.