

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.  
**Answer: TRUE.**
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.  
**Answer: TRUE.**
3. The Simplex algorithm runs in polynomial time.  
**Answer: FALSE.**
4. If every capacity in a flow network is integral then the maximum flow between any two nodes is integral.  
**Answer: TRUE.**
5. For any random variable  $X$ ,  $\mathbb{P}[(X - \mathbb{E}[X])^2 \geq t^2] \leq \mathbb{V}[X]/t^2$ .  
**Answer: TRUE.**

**Question 2 (Numbers):** Let  $A = \{a_1, a_2, \dots, a_n\}$  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  $A$ :  $O(n \log n)$ .
  - (b) Computing  $\max(A)$ :  $O(n)$ .
  - (c) Computing  $\text{median}(A)$ , i.e., the element with rank  $(n + 1)/2$ :  $O(n)$ .
  - (d) Computing  $\sum_{1 \leq i < n} (a_{i+1} - a_i)$ :  $O(1)$  since this simplifies to  $a_n - a_1$ .
  - (e) Computing  $\sum_{1 \leq i < j \leq n} (a_i - a_j)^2$ :  $O(n)$  since this simplifies to  $\sum_i a_i^2 - (\sum_i a_i)^2$ .
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  $\{a_1, \dots, a_i\}$  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.  
**Answer:**

$$M[i, j] = \begin{cases} 1 & \text{if } a_i = j \\ M[i - 1, j] & \text{if } a_i > j \\ \max\{M[i - 1, j], M[i - 1, j - a_i]\} & \text{if } a_i < j \end{cases} .$$

The reasoning is  $M[i, j] = 1$  iff  $a_i = j$  (i.e., the  $i$ th element has exactly the correct size) or  $M[i - 1, j] = 1$  (i.e., there was already a subset of  $\{a_1, \dots, a_{i-1}\}$  that sums to  $j$ ) or there was a subset of  $\{a_1, \dots, a_{i-1}\}$  that summed to  $j - a_i$  (since adding  $a_i$  ensures the sum is now  $j$ ).

- (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  $t$ ?

**Answer:**

- i. For all  $j$  let  $M[1, j] = 1$  if  $a_1 = j$  and  $M[1, j] = 0$  otherwise.
  - ii. For  $i = 2, \dots, t$ , update  $M[i, j]$  for all  $j$  using the above formula.
  - iii. Return  $M[n, t]$
- The running time is  $O(nt)$ .

### 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  $P = NP$ .

**Answer:** Suppose the NP-complete problem you find a polynomial time algorithm for is  $\Pi$ . Then, to solve any other problem  $\Pi'$  in NP in polynomial time, you can first reduce it to  $\Pi$  in polynomial time and then solve the resulting instance of  $\Pi$  using your algorithm.

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  $3\text{-SAT} \leq_P 2\text{-SAT}$ , i.e., any 3-SAT instance  $\phi$  can be transformed (in polynomial time) into a 2-SAT instance that is satisfiable iff  $\phi$  is satisfiable.

**Answer:** FALSE. If  $3\text{-SAT} \leq_P 2\text{-SAT}$  you could solve 3-SAT in polynomial time since 2-SAT can be solved in polynomial time. But since 3-SAT is NP complete, this would imply  $P = NP$ .

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

**Answer:** TRUE. Since  $P \subset NP$ ,  $2\text{-SAT} \in NP$ . But since 3-SAT is NP complete we already know all problems in NP can be reduced to it in polynomial time.

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

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

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

**Answer:** If  $C$  is satisfied then there exists a literal  $a_i$  that is true. Setting all  $y_i$  to TRUE if  $y_i$  appears to the left of  $a_i$  in  $f(C)$  and  $y_i$  FALSE otherwise, ensures  $f(C)$  is satisfied.

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

**Answer:** Suppose  $C$  is not satisfied but  $f(C)$  is satisfied. Then considering the first clause,  $y_1$  must be TRUE. Hence, considering the second clause,  $y_2$  must also be true. Continuing this logic, all  $y_i$  be must be true. But then the last clause of  $f(C)$  is not satisfied which is a contradiction.

**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  $t_1 \geq t_2 \geq \dots \geq t_n > 0$ . Your goal is to assign the tasks to the  $m$  people such that everyone finishes in the smallest amount of time. Let OPT be the smallest value such that it is possible to assign the tasks such that everyone finishes in  $\leq$  OPT time. For example, if  $n = 4$ ,  $m = 3$ ,  $t_1 = 4$ ,  $t_2 = 3$ ,  $t_3 = 2$ , and  $t_4 = 2$  then  $\text{OPT} = 4$  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  $t_1, \dots, t_n$ , explain why  $\text{OPT} \geq t_1$ ?

**Answer:** Since someone has to do task 1, that person can't be finished in less than  $t_1$  minutes.

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

**Answer:** Since the average amount of work everyone needs to do is  $\sum_{i=1}^n t_i/m$  there must be someone who does at least this amount of work.

3. For arbitrary  $t_1, \dots, t_n$ , explain why  $\text{OPT} \geq t_m + t_{m+1}$ ?

**Answer:** Since there must be someone who does two of the first  $m+1$  tasks  $\text{OPT} \geq t_i + t_j$  for some  $i < j \leq m+1$ . But this is at least  $t_m + t_{m+1}$ .

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 OPT minutes of work.

**Answer:** After the first  $m$  tasks have been assigned, person  $i$  is only doing task  $i$  and has therefore been assigned  $t_i \leq t_1 \leq \text{OPT}$  minutes of work.

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

**Answer:** When we assign the task  $j$  for  $j > m$ , a person is assigned the task only if they are currently assigned at most  $\sum_{i=1}^n t_i/m$  work. Hence, when job  $j > m$  is assigned a person's assignment increases to at most

$$t_j + \sum_{i=1}^n t_i/m \leq t_{m+1} + \text{OPT} \leq (t_m + t_{m+1})/2 + \text{OPT} \leq 1.5 \times \text{OPT} .$$

**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.

**Answer:** Suppose  $x, y \in U$  and there is an edge  $(x, y)$  in the original graph. Then note that  $x$  and  $y$  can't have been deleted in step 1. Hence the edge  $(x, y)$  still remains when we reach step 2. But then we are guaranteed to have deleted at least one of  $x$  or  $y$ .

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

**Answer:**  $\mathbb{E}[X] = np$  since there are  $n$  nodes and each is not deleted with probability  $p$ .

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

**Answer:**  $\mathbb{E}[Y] = mp^2$  since there are  $m$  edges and each is not deleted with probability  $p^2$  (since not deleting an edge requires that both its endpoints are not deleted).

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

**Answer:** First note that  $|U| \geq X - Y$  since at  $X$  nodes remain after step 1 and then at most  $Y$  of these nodes are deleted in step 2. Then, by linearity of expectation:

$$\mathbb{E}[|U|] \geq \mathbb{E}[X] - \mathbb{E}[Y] = np - mp^2 = \frac{n^2}{2m} - \frac{n^2m}{4m^2} = \frac{n^2}{4m}.$$

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.

**Answer:** Let  $X$  be the number of cliques of size  $r$  where every edge has the same color after we randomly 2-color the edges. Then  $\mathbb{E}[X] = \binom{n}{r} 2 \times 2^{-\binom{r}{2}}$  since there are  $\binom{n}{r}$  cliques of size  $r$  and each has its edges all of the same color with probability  $2 \times 2^{-\binom{r}{2}}$  (the extra factor of 2 is because there are two colors. But then

$$\binom{n}{r} < 2^{\binom{r}{2}-1} \implies \mathbb{E}[X] < 1,$$

and using the fact that  $X \in \{0, 1, 2, \dots\}$  we can deduce that  $\mathbb{P}[X = 0] > 0$ . Hence there exists a coloring where  $X = 0$ .