Princeton University
COS 521: Advanced Algorithms
Final Exam Fall 2008

Sanjeev Arora

Due Jan 14 5pm at the latest in my office, Room 307

Instructions: The test has seven questions. Finish the test within **48 hours** after first reading it. You can consult any notes/handouts from this class and feel free to quote, without proof, any results from there. You cannot consult any other source or person in any way.

DO NOT READ THE TEST BEFORE YOU ARE READY TO WORK ON IT.

Write and sign the honor code pledge on your exam (The pledge is "I pledge my honor that I have not violated the honor code during this exam and followed all instructions.")

I will be available Jan 3-13 by email to answer any questions. I will also offer to call you if your confusion does not clear up. In case of unresolved doubt, try to explain your confusion as part of the answer and maybe you will receive partial credit.

1. Assume you have a fair random coin. Show how to generate a random *permutation* from $\{1, 2, \ldots, n\}$ to $\{1, 2, \ldots, n\}$ using $O(n \log n)$ coin tosses and $O(n \log n)$ time.

2. Let $S_1, S_2, \ldots,$ be arbitrary distinct subsets of $\{1, 2, \ldots, n\}$. Suppose you randomly pick integer weights $w_1, w_2, \ldots, w_n$ from the range $[1, W]$ and let the weight of a subset S be $w(S) = \sum_{i \in S} w_i$. Show that the probability that the subset of lowest weight is unique is at least $1 - n/W$. (Hint: No fancy probabilistic reasoning is needed.)

3. We consider the problem of evacuating people out of a city that has been struck by a disaster (e.g., mega terrorism). We model the city's roads and bridges as weighted digraphs where the *capacity* of an edge is the number of cars that can traverse it in one hour. The starting location of each car is given.

   Give an efficient algorithm to determine the minimum amount of time needed to evacuate all people (= all cars) out of the city, where a car should not be forced to stop once it has been asked to move. The algorithm should be polynomial in the size of the network and in the number of bits used to represent the problem. (Clearly address this issue in your answer; also indicate any other assumptions you made.)

   Suppose the number of cars is far greater than the number of nodes in the network, and the input indicates the number of cars at each node at the start. Is your algorithm polynomial in this case? If not, can you make the algorithm polynomial?

   Aside: A rough calculation of this kind shows that evacuating New York City would take weeks.

4. Show that if there is a constant $\epsilon > 0$ such that the MIN VERTEX COVER problem can be approximated within a factor $1.5 - \epsilon$ then every 3-colorable graph can be colored with $O(\log n)$ colors.

5. Suppose you are given m halfspaces in $\Re^n$ with rational coefficients. Describe a polynomial-time algorithm to find the largest *sphere* that is contained inside the polytope defined by these halfspaces.

   For extra credit, try to design an algorithm for the case where the halfspaces are given implicitly using a separation oracle. (You may use the usual approximation notions here.)

6. In the k-center problem we are given n points $\{1, \ldots, n\}$ in a metric space with distance function $d(\cdot, \cdot)$ and a number k, and have to find a subset of k points S that minimizes

$$\max_{i \in \{1,\ldots,n\}} \min_{j \in S} d(i, j).$$

   Computing this is **NP**-hard. You have to give a 2-approximation for this problem.

7. A *DNF* formula consists of n variables and is the OR of m terms, where each term is an AND of literals. The goal in this problem is to design a fully polynomial approximation scheme for $\sharp DNF$, the problem of counting the number of satisfying assignments. Actually the algorithm is sketched below; you have to furnish any remaining details

and indicate how to choose N so that the estimate is correct up to $(1+\delta)$ multiplicative factor with probability at least $1-\epsilon$.

Note that an AND of literals has a unique satisfying assignment. Thus if a term has t literals, then 2^{n-t} assignments satisfy that term.

The algorithm is as follows. It defines N iid random variables $Z_1, Z_2, \ldots, Z_N$ where Z_i is defined as follows. First randomly pick a term, where a term with t literals is picked with probability proportional to 2^{-t}. Then pick an assignment uniformly at random from all assignments that satisfy this term. Then Z_i is ($\sharp$of terms satisfied by this assignment)$^{-1}$. The estimator is (up to scaling) the sum $\sum_i Z_i$.