# CS 671 (Fall 2020) Midterm (Takehome) — Due 10/23/2020 by e-mail

**This exam is takehome, but apart from having more time and getting to work on it wherever you want, you should treat it exactly like an open-book in-class exam. Specifically, here are the key rules. (If in doubt, check with the instructor.)**

**(a) You may access the textbook, anything I handed out or posted for this class (assignments, sample solutions, videos, whiteboard), and anything you wrote for this class (class notes, homework solutions). You may not access any other books, any online resources (not even Wikipedia), notes taken by classmates or friends, etc.**

**(b) You may not communicate about this exam with anyone except the instructor (whom you can ask by e-mail or set up a Zoom meeting with). Not with classmates, labmates, friends from undergraduate, or anyone else. Not even basic questions — you should communicate exactly as if you were sitting in an in-class exam, i.e., not at all.**

**(c) The exam is due by e-mail to David by 5:00pm Pacific on Friday, 10/23/2020. Contrary to homeworks, late submissions will not be accepted for the midterm.**

G o o d  L u c k

## Problem 1

This problem is about inferring an ordering of items from partial information. You are to assign an order to variables $x_1, \ldots, x_n$, so that each number from 1 to $n$ is assigned to exactly one variable. You are given $C$ constraints of the form "$x_j$ must lie between $x_i$ and $x_k$", written as $(x_i, x_j, x_k)$. This constraint is satisfied if either $x_i < x_j < x_k$, or $x_k < x_j < x_i$. Your goal is to satisfy as many such constraints as possible.

It is not difficult to come up with examples where not all $C$ constraints can be satisfied. It is also not terribly difficult to show that this problem is NP-hard to solve optimally. You are to give a constant factor approximation algorithm running in polynomial time (and prove its correctness and running time).

(a) Give a randomized algorithm that finds an ordering satisfying at least an $\alpha \leq 1$ fraction of the number of constraints that the optimum solution satisfies, in expectation[1]. The constant $\alpha$ should be independent of $n$ and $C$.

(b) Derandomize your algorithm using the method of conditional expectations, resulting in a deterministic algorithm with the same approximation guarantee that does not make any reference to "expectation", "probability" or any other probabilistic terms. If your algorithm in the first part was already deterministic, there is nothing to do in this part. (This part is not easy.)

## Problem 2

Consider a random multigraph $G$ generated from the configuration model in which node $v$ has degree $d_v$. (Reminder: this means that we draw a uniformly random multigraph — allowing self-loops and parallel edges — subject to the requirement that node $v$ must have exactly $d_v$ edges. A self-loop counts as two edges. Different $v$ may have different $d_v$ values.) Now, suppose that $V$ is partitioned disjointly into sets $S_1, S_2, \ldots, S_k$, not necessarily randomly, but before the graph is generated. Calculate the expected number $X_{ij}$ of edges between $S_i$ and $S_j$, and show that $X_{ij}$ is, with high probability, concentrated to within $o(m)$ of this expectation, where $m = \frac{1}{2} \sum_v d_v$ is the total number of edges in the graph.

[1] If you find a deterministic constant-factor approximation algorithm, that would also be acceptable.

## Problem 3

Suppose that you are given a set of $n$ points in the plane, each representing a patient (the two coordinates could be, say, blood pressure and cholesterol level). Each patient is to be classified as either high-risk for some disease, or low-risk. You are told that there is some line in the plane such that all patient on one side of the line are high-risk, and those on the other side are low-risk. Your goal is to find that line.

In order to help you find the line, you can run extensive tests on a patient, and find out whether he is actually high-risk or low-risk. The test result is always correct. However, as these tests take a lot of time and effort, you want to minimize the number of such tests you perform, and still classify most patients correctly as high-risk or low-risk. In order to save some tests, you are willing to misclassify a small fraction of patients.

Here is what your algorithm is to do. It is given the locations of all $n$ points, but without their "high-risk"/"low-risk" labels. It is also given parameters $\epsilon$ and $\delta$. It next runs tests on a number of patients depending only on $\epsilon$ and $\delta$, but not on $n$. Based on the outcomes of those tests ("high-risk" or "low-risk" for each tested patient), it next classifies all untested patients as "high-risk" or "low-risk", in time polynomial in $1/\epsilon, 1/\delta$, and $n$. Here is the guarantee that your algorithm should satisfy: The number of patients that are labeled incorrectly should be at most $\epsilon n$, with probability at least $1-\delta$.

Give a randomized algorithm with these guarantees, and prove its correctness.

## Problem 4

Remember that the Hamiltonian Path problem is the following: you are given a graph $G$ (undirected, for our purposes), and are to decide if there exists a path in $G$ that visits each node exactly once. You probably also remember that this problem is NP-complete. So we suspect that there is no polynomial-time algorithm (deterministic or randomized) for this problem.

Here, we are aiming for a much less ambitious lower bound, showing that each randomized algorithm for this problem must take at least $\Omega(n^2)$ steps on an $n$-node graph. More specifically, we assume that the graph is given by an oracle which we can ask, for any pair $(u, v)$, whether there is an edge between $u$ and $v$. We only count the number of such questions the algorithm asks the oracle, and give the algorithm all other computation for free.

Prove that any randomized Las Vegas algorithm that always decides correctly whether the graph has a Hamiltonian Path must ask at least $\Omega(n^2)$ questions of the oracle in the worst case.