

CS599 (Spring 2009) - Takehome Midterm

Due in David's office by noon on Friday, 03/27

No late submission will be granted, so you may want to start early. Contrary to the policy on homeworks, you cannot discuss problems, ideas, or solutions to this midterm with anyone (in our outside of class) except the instructor. Also, as a reminder, you are not allowed to seek solutions to these problems online or elsewhere. If you are stuck on a problem and need hints, or you need help understanding a problem, you are welcome to e-mail me, or talk to me in person.

Problem 1

The first problem is about inferring an ordering of items from partial information. You are to assign an order to variables x_1, \dots, 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_i must lie between x_j and x_k ”, written as (x_i, x_j, x_k) . This constraint is satisfied if either $x_j < x_i < x_k$, or $x_k < x_i < x_j$. 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). That is, 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 expectation¹. The constant α should be independent of n and C .

Problem 2

Suppose that you are organizing a soccer festival for a high school. There are a total of k students in the high school, and you want to play n soccer matches. Each match i will involve a set S_i of students. For instance, S_1 could be the set of all 10th graders, S_2 could be the set of all girls, S_3 could be the set of all left-handed people, and S_4 the set of all math wiz kids. Notice that these sets don't need to be disjoint. We assume that each set S_i has exactly the same size $2n$, i.e., twice as many students as there are matches total.²

When you play a match among S_i , you need to divide S_i into two teams, and those teams should be easy to distinguish. Thus, *before the festival starts*, you want to tell each student either to wear a red shirt, or to wear a blue shirt. Students wear the same shirt color for the entire day. Then, when you play the match for S_i , you simply have all red students in S_i play against all blue students in S_i .

Of course, if you choose a bad assignment, the teams in one or more of the matches could be badly balanced in size. It is easy to come up with examples in which it is not possible to make all games be balanced. Instead, we strive to have them nearly balanced. We say that the match for S_i is *nearly balanced* if the size of the larger team is at most twice as big as the size of the smaller team. Give an algorithm (randomized or deterministic) for finding a shirt assignment to students such that all matches are nearly balanced under your shirt assignment. Prove that your algorithm is correct, and analyze its running time. Your algorithm should run in (expected) polynomial time.

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.

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

²Yes, this is somewhat arbitrary.

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 ϵ and δ . It next runs tests on a number of patients depending only on ϵ and δ , 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 ϵn , 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.