# 15-854 Approximation and Online Algorithms

**Take-home final** **Due: 24 hours after you begin**

**Groundrules:** You can use the book, class notes and handouts. If you need to consult an elementary probability text, that is fine too. You should work alone. Send me email if questions are confusing or unclear. If necessary, I will post an errata on the web page.

This take-home final is worth between 1 and 3 typical homework assignments: at the top of your hand-in, write down 1, 2, or 3, indicating how many you would like yours to be worth.

---

1. We are given a set of linear equalities modulo a prime $p$ (e.g., if $p = 3$ we might have $2x_1 + x_2 + 2x_5 \equiv 1$ and $x_1 + x_3 \equiv 2$). Assume that each equality has at least one variable in it.

   (a) Give a very simple randomized algorithm that will find an assignment to the variables that satisfies an expected $1/p$ fraction of the equalities.

   (b) Give a deterministic algorithm that will find an assignment that satisfies at least a $1/p$ fraction of the equalities.

2. In class, we saw how to get an $O(\log n)$ approximation to the "minimum ratio cut" problem. That is, given a graph $G$, we want to find a subset of vertices $S \subset V$ that minimizes the ratio (# edges between $S$ and $V - S)/(|S| \cdot |V - S|)$.

   (a) Show how you can use this result to find a 1/3-2/3 separator (a split such that $1/3 \leq |S| \leq 2/3$) with a number of edges that is within $O(\log n)$ of the number of edges in the best bisection (a split such that $|S| = 1/2$). Please do this without looking at the proof on p.187 of the Cheriyan-Ravi notes.

   (b) Give an example of a graph with a very small 1/3-2/3 separator but that doesn't have a small bisection.

3. Suppose we are given a 3-colorable graph and want to split it into two pieces such that neither piece has a triangle (a clique of size 3).

   (a) Describe a way of doing this using semi-definite programming.

   (b) Here is a method along the lines of the "faster exact 3-SAT" algorithm presented in class. First, split the graph arbitrarily into two pieces. Then, until we are done, repeat the following: pick an arbitrary triangle in one of the pieces, pick one of the three vertices of that triangle *at random*, and move that vertex to the other side.

   Prove that with probability at least 3/4, this algorithm halts in $O(n^2)$ steps. Hint: it may help to think of each of the two pieces as a "color" and identify those colors with two of the three colors used in some correct 3-coloring.

4. We saw in class that no deterministic online algorithm for paging can have a competitive ratio less than $k$. ($k$ is the number of pages that fit into fast memory.)

   What if we allow our algorithm to have *lookahead*? Specifically, for some $\ell > 0$, consider an online algorithm that gets to see $\ell$ requests into the future when making each decision. For instance, if $\ell = 2$ and the request sequence was $(1, 2, 3, 3, 2, 3)$, then at time 1 the algorithm gets to see $(1, 2, 3)$; at time 2, the algorithm gets to see $(2, 3, 3)$, and so on. Can a deterministic online algorithm for paging with lookahead get competitive ratio $< k$? Why or why not.

5. In this problem we will consider the 1-server problem with lookahead $\ell = 1$, but with a twist. The twist is that the online algorithm is allowed to visit the requests out-of-order. Specifically, at any point in time, the online algorithm has two requests $\{r_0, r_1\}$ and makes a decision to visit one of them. When it reaches that point, it then is told a new point $r_2$ to visit (so its new set is either $\{r_0, r_2\}$ or $\{r_1, r_2\}$ depending on whether it chose to visit $r_1$ or $r_0$. You can think of this as a traveling salesman who begins with a list of two cities to visit and, each time he travels to some destination, calls up the home office to get a new city. At the end of the game, the algorithm is told there is no new request and then it must visit the one remaining point (city) on its list.

   What is the competitive ratio of the algorithm that simply visits the requests in order (ignoring its lookahead)? By "competitive ratio" we mean that we are comparing its performance to the best it could have done in hindsight, had it made the correct decisions; e.g., if $\{r_0, r_1\}$ are the only two requests visible, we are allowing the offline algorithm to visit either $r_0$ or $r_1$ first, but are *not* allowing the offline algorithm to visit $r_2$ first. Prove both lower and upper bounds; that is, give an example where this algorithm does as badly as the ratio you give, and prove that it never does worse.

Extra. You can use this problem in place of one of the other problems on this test if you wish. If so, state which other problem is being replaced by this one.

Recall the rent-or-buy problem. If the rent cost is 1 and the buy cost is $B$, then the optimal deterministic strategy was to rent for $B - 1$ times and then buy. This gives a competitive ratio of 2 (in the limit as $B \to \infty$). Prove that with randomization, it is possible to achieve $e/(e - 1)$.