Homework 4

Out: *Nov 8* Due: *Nov 22*

Instructions:

- Upload your solutions (to the non-extra-credit) to each problem as a **separate PDF** file (one PDF per problem) to codePost. Please make sure you are uploading the correct PDF! Please anonymize your submission (i.e., do not list your name in the PDF), but if you forget, it's OK.
- If you choose to do extra credit, upload your solution to the extra credits as a single separate PDF file to codePost. Please again anonymize your submission.
- You may collaborate with any classmates, textbooks, the Internet, etc. Please upload a brief "collaboration statement" listing any collaborators as a separate PDF on codePost (if you forget, it's OK). But always **write up your solutions individually**.
- For each problem, you should have a solid writeup that clearly states key, concrete lemmas towards your full solution (and then you should prove those lemmas). A reader should be able to read any definitions, plus your lemma statements, and quickly conclude from these that your outline is correct. This is the most important part of your writeup, and the precise statements of your lemmas should tie together in a correct logical chain.
- A reader should also be able to verify the proof of each lemma statement in your outline, although it is OK to skip proofs that are clear without justification (and it is OK to skip tedious calculations). Expect to learn throughout the semester what typically counts as 'clear'.
- You can use the style of Lecture Notes and Staff Solutions as a guide. These tend to break down proofs into roughly the same style of concrete lemmas you are expected to do on homeworks. However, they also tend to prove each lemma in slightly more detail than is necessary on PSets (for example, they give proofs of some small claims/observations that would be OK to state without proof on a PSet).
- Each problem is worth twenty points (even those with multiple subparts), unless explicitly stated otherwise.

Problems:

§1 Consider the following variant of online set cover (slightly different than what we saw in class — keep an eye on the details). Offline, we are given a universe $U := \{1, \dots, n\}$ of n elements and a family $\mathcal{S} := \{S_1, \dots, S_m\}$ of m sets where each $\bigcup_i S_i = U$. The algorithm starts with $A = \emptyset$ which denotes the collection of selected sets.

In each time step $t \in \{1, \ldots, T\}$, an adversary reveals an element $e_t \in U$, and the online algorithm has to immediately ensure that $e_t \in \bigcup_{S \in A} S$, i.e., if e_t is already covered then the algorithm doesn't need to select a new set, and otherwise the algorithm has to select a set into A that contains e_t. The goal of the algorithm is to minimize the size of A.

To be clear: it may be that not all elements of U are eventually revealed.

Show that there are problem instances such that the offline optimal solution has $|A| = O(1)$ but any deterministic online algorithm has size $\Omega(\log(mn))$, which implies that no deterministic online algorithm can have $o(\log(mn))$ competitive ratio.

Hint: Think of instances where m and n are polynomially-related, so that $\log n = \Theta(\log m)$.

Remark: The $\Omega(\log(mn))$ bound also holds against randomized online algorithms, but you do not have to prove this.

§2 Given black-box access to a poly-time algorithm $\mathcal{A}_P$ that optimizes linear functions over the convex, compact region P, and poly-time $\mathcal{A}_Q$ that optimizes linear functions over the convex, compact region Q, design a poly-time algorithm that optimizes linear functions over the convex, compact region $P \cap Q$.

Note: For this problem, you do *not* need to check bounding boxes, bit complexity, etc. For example, you may assume that whenever you have a separation oracle for a convex, compact region R that the Ellipsoid algorithm optimizes linear functions over R in poly-time.

§3 Define a *corner* of a convex, compact region P to be any $x \in P$ that cannot be written as a convex combination of other points in P.[1] Given black-box access to a poly-time algorithm $\mathcal{S}_P$ that is a separation oracle for convex polytope[2] $P \in \mathbb{R}^n$, design a poly-time algorithm that takes as input a point x and writes x as a convex combination of corners of P. That is, output a list $\{(c_1, y_1), \ldots, (c_{n+1}, y_{n+1})\}$ such that each y_i is a corner of P, each $c_i \geq 0$, and $\sum_i c_i = 1$.

Note: For this problem, you do *not* need to check bounding boxes, bit complexity, etc. For example, you may assume that whenever you have a separation oracle for a convex, compact region R that the Ellipsoid algorithm optimizes linear functions over R in poly-time. Moreover, you may assume that $\mathcal{S}_P$ outputs a separating hyperplane that is a *facet* of P. For example, if P is the unit hypercube, you may assume that $\mathcal{S}_P$ always outputs a separating hyperplane of the form $x_i = 0$ or $x_i = 1$, for some i.

§4 In the submodular welfare problem, there are n bidders and m items. The value of bidder $i \in \{1, \ldots n\}$ for a subset of items $S \subseteq \{1, \ldots, m\}$ is given by a monotone submodular function $f_i(S)$ where $f_i(\emptyset) = 0$. We want to allocate the m items to the

[1] So for example, if P is a triangle, P has three corners. If P is a circle, it has infinitely many.

[2] Recall that a convex polytope can be written as the intersection of finitely-many halfspaces. Alternatively, it is the convex hull of finitely-many points.

n bidders, i.e., find an item partition where bidder i gets susbset $S_i \subseteq \{1, \dots, m\}$ and $S_i \cap S_j = \emptyset$ for $i \neq j$, and the goal is to maximize the welfare $\sum_{i \in \{1,\dots,n\}} f_i(S_i)$. Show that the following simple greedy algorithm gives a 2-approximation:

(a) Initialize $S_i = \emptyset$, for all bidders i.

(b) For item $j = 1$ to m:

i. Let $i_j := \arg\max_i \{f_i(S_i \cup \{j\}) - f_i(S_i)\}$. That is, let i_j be the bidder who gets greatest marginal value for adding item j to their current set S_i. Break ties arbitrarily (but pick exactly one arg max).

ii. Add item j to S_{i_j}, leave all other S_{i_j} unchanged.

(c) Output $S_1, \dots, S_n$.

Hint: Note that a submodular function remains submodular even if you "contract" a set, i.e., $f_S(A) := f(S \cup A) - f(S)$ is also a submodular function on elements $\{1, \dots, m\} \setminus S$.

§5 Design a *randomized* communication protocol for Equality. That is, assume that Alice and Bob have access to an infinite stream of shared random bits (and accessing these bits doesn't count towards the communication of the protocol). Design a communication strategy where Alice and Bob each output only $O(1)$ bits, such that:

- If Alice and Bob have equal inputs, they will certainly output "yes."
- If Alice and Bob have unequal inputs, they will output "no" with probability at least $2/3$ (where the probability is over the randomness in the shared random stream).

Extra Credit:

§1 (Extra Credit) Consider the following variant on the secretary problem: an adversary puts the elements into any order they desire. Then, instead of being randomly permuted, the elements are revealed either in order, or in reverse order, each with probability $1/2$ (everything else is the same: upon seeing an element, you must immediately and irrevocably accept or reject). Prove that no algorithm can guarantee acceptance of the heaviest element with probability $> 1/n$ when there are n elements.

§2 (Extra Credit) Consider the following variant on prophet inequalities: instead of each X_i being independently drawn, there is a joint distribution over $(X_1, \dots, X_n)$ (everything else is the same: you know the joint distribution, the random variables X_i are revealed to you in order, and you must immediately accept/reject upon seeing). Prove that no algorithm can guarantee better than $\mathbb{E}[\max_i X_i]/n$.

§3 (Extra Credit) A *non-deterministic* communication protcol for $f(\cdot, \cdot)$ has the following properties (similar to non-deterministic algorithms):

- Alice decides what to say in round i deterministically as a function of her own input, $A \in \{0,1\}^n$, an advice string, S, and the transcript during rounds 1 thru $i - 1$.

- Bob decides what to say in round i deterministically as a function of his own input, $B \in \{0,1\}^n$, an advice string, S, and the transcript during rounds 1 thru $i-1$.
- If $f(A,B) = 1$, then there exists an advice string S such that Alice and Bob will output 1.
- If $f(A,B) = 0$, then for all advice strings S, Alice and Bob will output 0.
- $|S|$ counts towards the amount of communication.

a) Design a non-deterministic algorithm for NotEquality (i.e. $f(A,B) = 1$ if and only if $A \neq B$), and another for NotDisjointness (i.e. $f(A,B) = 1$ if and only if $A \cap B \neq \emptyset$), each using total communication $O(\log n)$.

b) Prove that every non-deterministic algorithm for Equality and Disjointness require communication n.

c) For $f : \{0,1\}^n \times \{0,1\}^n \Rightarrow \{0,1\}$, let $\bar{f}(\cdot,\cdot) := (1-f)(\cdot,\cdot)$ (that is, $\bar{f}(x,y) := 1 - f(x,y)$). Let $D(f), ND(f)$ denote the optimal deterministic and non-deterministic communication complexity for f, respectively. Prove that:

$$D(f) = O(ND(f) \cdot ND(\bar{f})).$$

Hint (Part c): Recall the concept of a 1-rectangle from lecture, a set of inputs S for Alice and T for Bob such that $f(x,y) = 1$ for all $x \in S, y \in T$. Say that a collection $(S_1,T_1),\ldots,(S_k,T_k)$ of 1-rectangles covers f if $\cup_i\{(x,y)|\ x \in S_i, y \in T_i\}$ contains *every* (x,y) such that $f(x,y) = 1$. First, try to draw a connection between $ND(f)$ and the minimum k such that a collection of k 1-rectangles covers f (and do the same for $ND(\bar{f})$ and 0-rectangles).