Princeton Univ Fall '25	COS 521: Advanced Algorithms
Homework 4	
Out: *Oct 29*	Due: *Nov 11*

Instructions:

- Upload your non-extra solutions to Gradescope in a single PDF file, and mark your solution to each problem. 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 Gradescope. 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 Gradescope (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 In class, we showed that by doing a binary search on the optimal value of an LP, optimizing the objective value can be reduced to deciding feasibility, which is then solvable by the Ellipsoid algorithm. Show how to modify Ellipsoid so that it solves the optimization problem directly, i.e., this reduction is in fact not necessary.

Hint: Maintain the invariant that the optimal solution is contained in the current ellipsoid.

Note: For this problem, you do *not* need to worry about bounding boxes, bit complexity, precision, etc. For example, it suffices that the algorithm finds some x that is $2^{-\Theta(n)}$-close to the true optimum in distance. Also, you do *not* need to worry about the feasible region having dimension less than n, and you may assume the volume of the feasible region is at least $2^{-\Theta(n)}$. You also don't need to reprove the lemmas used in Ellipsoid.

§2 The Ellipsoid algorithm we saw in the lecture solves convex programs assuming a separation oracle. Here, we want to show the opposite. To be more specific, consider the following two tasks regarding a convex body $\mathcal{K}$:

- $\mathsf{OPTIMIZE}_{\mathcal{K}}$: given a vector $c \in \mathbb{R}^n$, output $\arg\max_{x \in \mathcal{K}} c^\top x$;
- $\mathsf{SEPARATE}_{\mathcal{K}}$: given a point $x \in \mathbb{R}^n$, output either $x \in \mathcal{K}$, or a separating hyperplane.

We are going to show that if for a specific convex body $\mathcal{K}$, there is a polynomial time algorithm for $\mathsf{OPTIMIZE}_{\mathcal{K}}$, then there is a polynomial time algorithm for $\mathsf{SEPARATE}_{\mathcal{K}}$.

(a) Suppose for a given x, we can solve the following LP with *infinitely many* constraints (finding the optimal w and T). Show that then we can solve $\mathsf{SEPARATE}_{\mathcal{K}}$.

$$\begin{aligned} \text{Variables: } & w \in \mathbb{R}^n, T \in \mathbb{R} \\ \text{Maximize: } & w^\top x - T \\ \text{Subject to: } & \forall y \in \mathcal{K}, w^\top y \le T \\ & -1 \le T \le 1 \end{aligned}$$

(b) Design a polynomial time separation oracle for the above LP using $\mathsf{OPTIMIZE}_{\mathcal{K}}$, and conclude.

§3 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 may assume that both P and Q are bounded (otherwise, there are some directions in which it is not possible to optimize linear functions because the optimum is $\pm\infty$). But, you should not assume that you know the bounding box containing them (if you want to access this, you should describe how to find it).

Aside from this, you do not need to worry about any other "technical" aspects of running the Ellipsoid algorithm. That is, you may assume for the sake of this problem that if you have a bounding box B containing a region R, and a separation oracle for the region R, that you can optimize linear functions over R in poly-time.

§4 Let N be a set of size n, and $f : 2^N \to \mathbb{R}$ be a function on all subsets of N. Show that if f is *not* submodular, then its Lovász extension $\hat{f}$ is *not* convex.

§5 In the submodular welfare problem, there are n bidders and m items. The value of bidder $i \in \{1, \dots n\}$ for a subset of items $S \subseteq \{1, \dots, m\}$ is given by a monotone submodular function $f_i(S)$ where $f_i(\emptyset) = 0$. We want to allocate the m items to the n bidders, i.e., find an item partition where bidder i gets subset $S_i \subseteq \{1, \dots, m\}$ 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 $\{1, \dots, m\} \setminus S$. You may use this fact without proof.

Extra Credit:

§1 Consider the following variant of online set cover. 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 $\bigcup_i S_i = U$. The algorithm starts with $A = \emptyset$ which denotes the collection of selected sets.

In each time step $t \in \{1, \dots, T\}$, 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 for $n \leq m$, every (possibly randomized) algorithm achieves an expected competitive ratio of at best $\Omega(\log(n))$.