Usual rules. Please write the names of your collaborators (if any) on the HWs. Solve any three of the five problems.

**Exercises**

1. **Langrangian Duality for Linear Programs.** Use the recipe from the blog post on Lagrange multipliers to write down the Lagrange dual of the linear program

   \[
   \min \{ c^\top x \mid Ax \geq b, x \geq 0 \}.
   \]

   Observe that the “best lower bound” you get this way is the usual linear programming dual.

2. **Misc Approximation Algorithms.**
   
   (a) Consider the following randomized algorithm for vertex cover with vertex weights: Start with \( C = \emptyset \) and enumerate over the edges of \( G \). If the current edge \( \{u, v\} \) is covered, move on to the next edge, else pick one of the endpoints as follows: pick \( u \) w.p. \( \frac{w_u}{w_u + w_v} \), and pick \( v \) otherwise. Show that \( E[w(C)] \leq 2OPT \).

   (b) Consider the \( k \)-dispersion problem: given a metric space \( (V, d) \) and a value \( k \), find a set \( S \subseteq V \) with \( k \) points to maximize \( \min_{i,j \in S} d(i,j) \). Here’s one algorithm: start with an arbitrary point \( s_1 \in V \). Now at each step pick a point

   \[ s_i = \arg \max_{x \in V} \{d(x, \{s_1, s_2, \ldots, s_{i-1}\})\}. \]

   Show that the algorithm is a \( \frac{1}{2} \)-approximation.

   (c) Consider now a very related problem, called \( k \)-center. Same setting, pick \( S \) with \( |S| = k \) to minimize \( \max_{i \in V} d(i, S) \). Show that the same algorithm above is a 2-approximation.

3. **Sums of Squares.**
   
   (a) Show that the famous Motzkin polynomial

   \[ P(x, y) = x^4y^2 + x^2y^4 + 1 - 3x^2y^2 \geq 0 \]

   for all \( x, y \). (Hint: AM-GM inequality.) Moreover, using elementary arguments show that \( P \) is not SOS. (You won’t need duality for this fact.)

   (b) Show that the polynomial

   \[ P(\bar{x}) = x_1^4 - (2x_2x_3 + 1)x_1^2 + (x_2^2x_3^2 + 2x_2x_3 + 2) \]

   is SOS.

4. **Many Prophets.** Consider a variant of the linear program we used to analyze the prophet inequality problem, where now you are allowed to pick up to \( k \) items:

   \[
   \max \sum_{i, v} y_{i,v} \cdot v \\
   \sum_{i, v} y_{i,v} \leq k \\
   y_{i,v} \in [0, p_{i,v}]
   \]
Use the rounding scheme where you consider the items in order. If the r.v. $X_i$ takes on value $v$ (which happens with probability $p_{iv}$) then pick it with probability $(1 - \delta) \frac{y_{iv}}{p_{iv}}$ for some $\delta \ll 1$ to be chosen later. Show that the expected value you get from r.v. $X_i$ is at least

$$\Pr\text{[you reach } i \text{]} \cdot (1 - \delta) \cdot v \cdot y_{iv}.$$ 

Moreover, show that you will reach $i$ with probability at least $1 - \exp(-c\delta^2 k)$. Hence infer that if you set $\delta = c' \sqrt{\frac{\log k}{k}}$, you will get expected value at least $LP \cdot (1 - c'' \sqrt{\frac{\log k}{k}})$. Here $c, c', c''$ are all constants.

**Problems**

1. **Solving Very Large LPs.** We saw that the Ellipsoid algorithm can solve LPs in polynomial time, even if they have an exponential number of constraints, as long as we have a strong separation oracle for the LP. What if there are a large number of variables but a small number of constraints? In some cases, Ellipsoid can help here too.

   In the parts below, we consider the bin-packing problem with $n$ items $[n] = \{1, 2, \ldots, n\}$, where the item sizes $s_i$ are positive integers, and the bin size $B$ is poly($n$).

   (a) Recall that a configuration is a subset $C \subseteq [n]$ such that $s(C) := \sum_{i \in C} s_i \leq B$. Suppose, for some reason, each item $i$ also has a value $v_i \in \mathbb{R}_{\geq 0}$, show how to find the max-value configuration in poly($n$) time. Your algorithm should work even if the values are not integers or bounded by poly($n$). (Hint: DP.)

   (b) (Do not submit.) Consider a simplification of the bin-packing LP from lecture, with a variable $x_C$ for each configuration $C$ (saying whether we chose configuration $C$ or not).

   $$\begin{align*}
   \min \sum_C x_C \\
   \sum_{C : i \in C} x_C \geq 1 & \quad \forall \text{ items } i \in [n] \\
   x_C \geq 0 & \quad \forall \text{ configurations } C.
   \end{align*}$$

   Show that the dual for this LP is:

   $$\begin{align*}
   \max \sum_{i \in [n]} y_i \\
   \sum_{i \in C} y_i \leq 1 & \quad \forall \text{ configurations } C \\
   y_i \geq 0 & \quad \forall \text{ items } i \in [n].
   \end{align*}$$

   Let $OPT_{LP}$ denote the optimal solution value for these two LPs.

   (c) The strong separation problem for the dual is: given a purported solution $\hat{y} \in \mathbb{R}^n$, either correctly claim that $\hat{y}$ satisfies all the dual constraints, or output any one dual constraint that is not satisfied. Show how to solve this dual strong separation problem in time poly($n$).

   (d) Using this separation oracle, suppose Ellipsoid returns an optimal solution $y^* \in \mathbb{R}^n$ to the dual LP in poly($n$). (There is no dependence on the numbers, because they are all 0 or 1.) Recall that during its run, Ellipsoid looks at only some $m = \text{poly}(n)$ of the constraints (namely those which were returned by the strong separation oracle as being violated). Show that if $C^* = \{C_1, C_2, \ldots, C_m\}$ is the list of the configurations corresponding to
those constraints, such that \( y^* \) is also an optimal solution to the following poly(n)-sized LP, with \( \sum_i y^*_i = \OPT_{LP} \).

\[
\begin{align*}
\max & \sum_{i \in [n]} y_i \\
\sum_{i \in C} y_i & \leq 1 \quad \forall \text{ configurations } C \subseteq \mathcal{C}^* \\
y_i & \geq 0 \quad \forall \text{ items } i \in [n].
\end{align*}
\]

(Hint: what happens if run Ellipsoid again on the smaller LP? Will its behavior change?)

(e) Using duality again, show how to get a poly(n)-sized (primal) LP for bin-packing, which can be solved in poly(n) time, and whose solution can be extended to an optimal solution for the original primal LP (\( P \)).

To wrap up: if you have many constraints in your LP (but few variables), Ellipsoid works in poly(n) time if you can solve the primal separation problem. If you have many variables in your LP (but few constraints), don’t panic! You get an optimal solution to (P) in time poly(n), as long as you can solve the dual separation problem. This technique is essentially the same as column-generation, which is widely used in practice. The primal LP has an exponential number of columns and the dual separation oracle tells you which are the “interesting” columns to “generate” for your primal.

2. **A Max-Cut Like Problem.** Consider the problem

\[
\max_{x \in \{-1,1\}^n} x^T W x = \max_{x \in \{-1,1\}^n} \sum_{i,j} W_{ij} x_i x_j,
\]

for \( W \in \mathbb{R}^{n \times n} \) being a psd matrix. Let’s get a \( 2/\pi \) approximation for it.

(a) (Do not submit.) Show that if \( W \) is the Laplacian matrix for a graph, then this is just the max-cut problem.

(b) Consider the SDP \( \max \{ \sum_{i,j} W_{ij} \langle v_i, v_j \rangle \mid \|v_i\| = 1 \forall i \} \). Show that the random hyperplane rounding gives expected solution value

\[
\text{Alg} = \sum_{i,j} W_{ij} \text{sgn}(v_i \cdot g) \text{sgn}(v_j \cdot g).
\]

Here \( g \) is a \( n \)-dimensional Gaussian vector, where each entry is an independent draw from \( N(0,1) \). You may use the fact (proven in previous exercises) that \( g/\|g\| \) is a uniform random point on the surface of the unit sphere.

(c) Show the following identity: for any unit vectors \( a, b \), if \( g \) is a random Gaussian, then

\[
\frac{2}{\pi} \mathbb{E} \left[ \text{sgn}(g \cdot a) \text{sgn}(g \cdot b) \right] = a \cdot b + \mathbb{E} \left[ (g \cdot a - \sqrt{2/\pi} \text{sgn}(g \cdot a)) (g \cdot b - \sqrt{2/\pi} \text{sgn}(g \cdot b)) \right].
\]

Hint: by rotational symmetry, assume that \( a = a_1 e_1 + a_2 e_2 \) and \( b = e_1 \). Hence if \( g = (g_i)_{i=1}^n \), show that \( \mathbb{E}[(a \cdot g) \text{sgn}(b \cdot g)] = \mathbb{E}[a_1 g_1 \text{sgn}(g_1)] = \sqrt{2/\pi} a_1 \).

(d) Use the parts above to show that \( \frac{2}{\pi} \text{Alg} = \text{SDP} + \text{(blah)} \). Show that (blah) term is non-negative when \( W \) is PSD, and hence \( \text{Alg} \geq \frac{2}{\pi} \text{SDP} \geq \frac{2}{\pi} \OPT \).

3. **Small Set Cover.** In the set cover problem we’re given a set system \((U, \mathcal{S})\), with each set \( S \in \mathcal{S} \) having weight \( w_S \geq 0 \), and the goal is to pick some sets \( \mathcal{S}' \subseteq \mathcal{S} \) so that their union equals \( U \), and their total weight \( \sum_{S \in \mathcal{S}'} w_S \) is minimized. We have \( n := |U| \) elements and \( m := |\mathcal{S}| \) sets.

In lecture we solved the unweighted case using (i) a combinatorial analysis for greedy, and (ii) solving the LP optimally and then randomized rounding. Now suppose each set \( S \in \mathcal{S} \) has size at most \( B \leq n \), and sets have weights. We’ll show \( O(\log B) \)-approximations for the weighted setting.
(a) Show that the greedy algorithm is an $O(\log B)$-approximation even in the weighted case.

(b) For each element $e \in U$, let $S(e) \subseteq S$ be the lightest set that contains $e$. Show that
\[ \frac{1}{|S|} \sum_{e \in S} w_S(e) \leq OPT. \]

(c) Give a $O(\log B)$-approximation algorithm that solves the LP and then picks some of the sets (randomly) based on the optimal LP solution.

4. **Load Balancing with Picky Jobs.** You are given $n$ jobs (each job $j$ having size $p_j$) and $m$ machines, and you want to assign jobs to machines to minimize the makespan (the max load of any machine). But here’s the catch: not every job can be assigned to every machine. (E.g., some machines don’t have the power to work on some jobs, etc.)

In fact, there is a bipartite graph $(M, J, E)$, where $|J| = n$ are the jobs, $|M| = m$ are the machines, and there is an edge $(i, j) \in E$ if job $j$ can be assigned to machine $i$.

Let $OPT$ be the makespan of the optimal schedule.

(a) Show that for every pair of integers $m, n$ with $m \geq n$, there exists an instance where the greedy algorithm (which just considers the jobs in the given order $p_1, p_2, \ldots, p_n$) and assigns them to an arbitrary least loaded machine thus far) can result in a load of $\log_2 n \cdot OPT$.

(b) Hence we need a different algorithm. Here’s one: we write the following LP:

\[
\begin{align*}
\text{min} & \quad L \\
\text{subject to} & \quad \sum_{j \in J, (i,j) \in E} p_j x_{ij} \leq L \quad \forall i \in M \\
& \quad \sum_{i \in M, (i,j) \in E} x_{ij} = 1 \quad \forall j \in J \\
& \quad x_{ij} \geq 0
\end{align*}
\]

Let $(x^*, L^*)$ be an optimal (vertex) solution to this LP. Show that $L^* \leq OPT$.

(c) The support of $x^*$ is the set of edges $(i, j) \in E$ on which $x_{ij}^* \neq 0$. If $x^*$ is a vertex solution for the LP, show that its support cannot have any cycles.

(d) Since the support of $x^*$ is a forest $F \subseteq E$, for each tree $T \in F$ do the following: root $T$ at some vertex corresponding to a job. Now assign each job $j$ in $T$ to an arbitrary one of the machines which are its children in $T$. If $j$ has no child, then assign it to its parent machine.

Show that the total load of any machine $i$ after this process is at most
\[ L^* + \max_{j : x_{ij}^* \neq 0} p_j \leq 2OPT. \]

5. **Online Primal-Dual.** In the online (unweighted) set cover problem, you are given a set system $(U, S)$ in advance with $n$ elements and $m$ sets. You just don’t know which of the elements of $U$ you actually want to cover. An unknown sequence $\sigma$ of elements $e_1, e_2, \ldots, e_t$ arrives one by one: when an element $e_i$ arrives and is not already covered by the sets picked thus far by us, we have to pick a new set $S_{(i)}$ containing this element. (We are allowed to pick multiple sets $S_{(i,1)}, S_{(i,2)}, \ldots$ instead of just one set, if we deem fit to do so.)
Let \( \text{set}(\sigma) \subseteq U \) denote the set of elements that appear in \( \sigma \), and let \( \text{OPT}(\sigma) \) be the fewest number of sets in \( \mathcal{S} \) that cover the elements of \( \text{set}(\sigma) \) — note that \( \text{OPT} \) is purely an offline quantity. Suppose \( \mathcal{A} \) is an online algorithm for Set-Cover; let \( \mathcal{A}(\sigma) \) be number of sets picked by the algorithm when given the sequence \( \sigma \). (One can extend the definition of the problem to the weighted case, where each set has a cost \( c(S_i) \), and both \( \mathcal{A} \) and \( \text{OPT} \) now minimize the total cost, etc.) The goal is to develop an algorithm to minimize the competitive ratio: this is the smallest value \( \rho \) such that on every input sequence \( \sigma \), the ratio

\[
\frac{\mathcal{A}(\sigma)}{\text{OPT}(\sigma)} \leq \rho.
\]

If the algorithm \( \mathcal{A} \) is randomized, replace the numerator by \( \mathbb{E}[\mathcal{A}(\sigma)] \).

(a) **Example A:** Let \( n = 2^d \), the universe be \( \{0, 1\}^d \), and the set system \( \mathcal{S} = \{S_i\}_{i=0}^d \) with \( S_0 = \{0^d\} \), and for \( 1 \leq i \leq d \), \( S_i \) is the set of all strings with 1 in the \( i \)-th place. Given any deterministic algorithm \( \mathcal{A} \), show that the adversary can choose an input sequence \( \sigma \) such that \( \text{OPT}(\sigma) = 1 \) but \( \mathcal{A}(\sigma) = d \).

This shows that no deterministic algorithm for this problem has a competitive ratio of better than \( d = \log_2 n \).

(b) **Example B:** Let the universe be \( U = [n] \), and the set system be the collection of all subsets of \( U \) with \( \sqrt{n} \) elements (hence \( m = |\mathcal{S}| = \binom{n}{\sqrt{n}} \)). Given any deterministic algorithm \( \mathcal{A} \), show that the adversary can choose an input sequence \( \sigma \) such that \( \text{OPT}(\sigma) = 1 \) but \( \mathcal{A}(\sigma) = \sqrt{n} \).

This shows that no deterministic algorithm for this problem has a competitive ratio of better than \( \Omega(\frac{\log m}{\log n}) \).

(c) Suppose we want to cover the subset \( U' \subseteq U \). Recall the natural LP relaxation of the (unweighted) set cover problem, and its dual.

\[
\begin{align*}
\text{min} & \sum_{S \in \mathcal{S}} x_S \\
\text{s.t.} & \sum_{S \in \mathcal{S}} x_S \geq 1 \quad \forall e \in U' \\
x & \geq 0
\end{align*}
\]

\[
\begin{align*}
\text{max} & \sum_{e \in U'} y_e \\
\text{s.t.} & \sum_{e \in \mathcal{S}} y_e \leq 1 \quad \forall S \in \mathcal{S} \\
y & \geq 0
\end{align*}
\]

We will develop a primal-dual online algorithm to get an approximate fractional solution to this LP.

Recall that we don’t up front know the elements we want to consider: initially \( U' = \emptyset \), and when an element \( e \) arrives online, we get a new constraint (\( \sum_{S \in \mathcal{S}} x_S \geq 1 \)) in the primal, and a new variable \( y_e \) gets added to constraints in the dual for each \( S \ni e \).

We will construct (feasible) primal solution \( x \) and a (possibly infeasible) dual solution \( y \) such that \( \sum_{S} x_S \leq 2 \sum_{e} y_e \), and moreover \( y/O(\log m) \) is feasible. Explain why this proves that \( \sum_{S} x_S \) is at most \( O(\log m) \) times the optimal LP solution.

(d) Here is the procedure. Initialize \( x_S = 1/2m \) for all \( S \).

- When element \( e \) arrives, set \( y_e \leftarrow 0 \).
- While \( \sum_{S \ni e} x_S < 1 \) double all the \( x_S \) for sets \( S \ni e \); also \( y_e \leftarrow y_e + 1 \).

Show that \( \sum_{S} x_S \leq \frac{1}{2} + \sum_{e} y_e \) at all times. Show that after the first element has arrived, \( \sum_{S} x_S \leq \frac{3}{2} \sum_{e} y_e \).

(e) Show that, at any point in time, \( y/(\log_2 2m) \) is a feasible dual solution.
(f) We just showed how to obtain a fractional solution to the minimum set cover which has value at most $O(\log m)$ times the optimal LP solution. Using randomized rounding (if an uncovered element $e$ arrives and causes sets $S \ni e$ to be doubled for zero or more times, at the end of the process, independently pick each set $S \ni e$ with probability $O(\log n) \cdot x_S$), show that the expected cost of this process is at most $O(\log n)$ times the online LP solution, and hence at most $O(\log n \log m)$ times the optimal number of sets to cover the elements in the input sequence so far.