Homework 3

Due Friday Dec 4, 2024.

Exercise 1.

(a) In Lecture we saw an algorithm for the facility location problem, which didn't require running a general linear program solver. It worked in two phases:

- Phase 1: Find a solution to the dual linear program. (This phase begins by initializing every α_j and β_{ij} to 0, and then starts growing the α_js. . . .)
- Phase 2: Round the dual solution to get a solution to the problem.

Show how to implement Phase 1 in $O(nm\log(nm))$ time, where n and m are the numbers of clients and facilities. The distances d_{ij} are given to your algorithm as a full $n \times m$ matrix.

(b) Consider an instance of the facility location problem where the distances between clients and facilities are either 1 or 3. Give a randomized rounding method that produces a solution with expected connection cost of $(1 + \frac{2}{e})C$ and expected facility cost of at most F + $f_{\min}\prod_i(1 - y_i)$ where F and C are the facility and connection costs of an optimal solution to the linear program relaxation used in class; y_i are the facility decision variables from that same program; and $f_{\min}$ is the least cost facility. (Our solution doesn't use the dual.)

Exercise 2.
Recall the basic setting of online learning. We have a finite set of actions $[n]$, and $T \geq n$ rounds. At each round t the player chooses a random action i_t according to some probability distribution over the set $[n]$ of actions. Simultaneously an adversary picks a loss function $\ell_t : [n] \to [-1, 1]$. At the end of the round the player only observes the loss of the action that she played, i.e. $\ell_t(i_t)$. For a strategy $\mathcal{S}$ that picks the distributions of $i_1, \ldots, i_T$, and loss functions $\ell_1, \ldots, \ell_T$, we define the regret with respect to the best action in hindsight as

$$R_T(\mathcal{S},\ (\ell_1, \ldots, \ell_T)) := \sum_{t=1}^{T} \mathbb{E}_{i_t \sim \mathcal{S}}\left[\ell_t(i_t)\right] - \min_{i\in[n]} \sum_{t=1}^{T} \ell_t(i).$$

The multiplicative weights strategy for picking the distributions of $i_1, \ldots, i_T$ over the the actions $[n]$ guarantees that

$$R_T(\text{Multiplicative Weights},\ (\ell_1, \ldots, \ell_T)) \leq 2\sqrt{T\log n},\ \text{for any loss function } \ell_1, \ldots, \ell_T.$$

Your task is to show that for any strategy of the player, there exists a sequence of loss functions that the adversary can pick such that the upper bound attained using multiplicative weights is (asymptotically) tight.

(a) For $Z_1, \ldots, Z_n \overset{\text{i.i.d.}}{\sim} N(0, 1)$ show that

$$\mathbb{E}\left[\max_{1\leq i\leq n} Z_i\right] = \Omega(\sqrt{\log n}).$$

Hint: You can use the following bounds for $Z \sim N(0, 1)$ and $x \geq 0$

$$\frac{1}{4}e^{-x^2} \leq \mathbb{P}(Z \geq x) \leq e^{-x^2/2}.$$

(b) Use the previous part and a CLT approximation in order to heuristically argue that: for any strategy $\mathcal{S}$ for picking the distributions of $i_1, \ldots, i_T$ over the actions $[n]$, there exists loss functions $\ell_1, \ldots, \ell_T$ such that

$$R_T(\mathcal{S},\ (\ell_1, \ldots, \ell_T)) = \Omega(\sqrt{T \log n}).$$

Hint: Define appropriate distributions over loss functions, and use an expectation argument.

Exercise 3.
Recall from lecture, the random walk on spanning trees in a graph is defined by a swap which consists of deleting a random edge and inserting a randomly chosen edge in the resulting cut. The state space or graph corresponding to this walk has a vertex for each spanning tree and an edge for each swap. Prove that from any spanning tree you can reach any other spanning tree in at most n swaps.

(See this paper to see how to extend this simple argument to give a bound on the "mixing time." The idea is that one embeds a complete graph in the state space where the congestion is small, as in multicommodity flow. And since a walk on the complete graph converges to the uniform distribution fast, one can derive an lower bound on the convergence rate for the random walk on the state space of spanning trees.

A general discussion of canonical paths and the mixing of markov chains is here.)

Exercise 4.
In class, we examined a random graph on n vertices and $n/8$ edges is not very connected. In this problem, you will show that a graph with n vertices and m edges is very connected for m a bit larger than $n/8$. A *bisection* is a partitioning of the vertices into two sets of equal size. You will show that the number of edges across every bisection is large with high probability.

For what it is worth, this is the idea behind the fact that random graphs are expanders.

The process of generating the graph is for each edge to choose two endpoints uniformly at random. Note that self-loops are possible.

Show that when $m = cn$, for large enough c, the number of edges crossing the *any* bisection of the vertices is $\Omega(n)$. (The expression $\frac{1}{2^m}\sum_{i\leq n}\binom{m}{n}$ might be useful with justification and bounding. It might also be useful to notice that the number of bisections is less than 2^n and use some bounds from the power of two choices lecture.)

Exercise 5.

One of the basic problems in relational databases is computing the size of the join of two relations. Recall that for two relations (i.e., tables in a database) $r(A, B)$ and $s(A, C)$, with a common attribute (i.e., column) A, we define the join $r \times s$ to be a relation consisting of all tuples (a, b, c) such that $(a, b) \in r$ and $(a, c) \in s$ Therefore if $f_r(a)$ is the number of occurrences of a in the A column of relation r (resp. $f_s(a)$), then the size of the join is $\sum_a f_r(a) f_s(a)$. Give an efficient streaming algorithm for estimating the size of the join. i.e. scan the items of two relations in a single pass in streaming fashion to estimate the size of the join. (The error can involve $\sum_a f_r(a)^2$, $\sum_a f_s(a)^2$, $\sum_a f_r(a) f_s(a)$. Specifically, it should be at most $O(\max\{\sum_a f_r(a)^2, \sum_a f_r(b)^2, \sum_a f_r(a) f_s(a)\})$.)