Homework Out: March 2
Due Date: March 16, midnight

The HW contains some exercises (fairly simple problems to check you are on board with the concepts; dont submit your solutions), and problems (for which you should submit your solutions, and which will be graded). Some problems have sub-parts that are exercises. For this problem set, its OK to work with others. (Groups of 2, maybe 3 max.) That being said, please think about the problems yourself before talking to others. Please cite all sources you use, and people you work with. The expectation is that you try and solve these problems yourself, rather than looking online explicitly for answers. Submissions due at beginning of class on the due date. Please check the Piazza for details on submitting your *LaTeXed* solutions.

Exercises

1. **Flows, Kings, Halls, and Colors.** Recall that Konig's theorem says for a bipartite graph G, the size of the maximum cardinality of a matching in G is equal to the size of the minimum vertex cover.

   (a) Use the max-flow/min-cut theorem to prove Konig's theorem. (Recall that the max-flow/min-cut theorem says that in any flow network, the maximum $s-t$ flow equals the minimum $s-t$ cut. Moreover if the arc capacities are integers, then there is a max flow which is integral.)

   (b) Use Konigs theorem to prove Hall's theorem:
   In a bipartite graph $G = (L, R, E)$, for any set $S \subseteq L$, let $N(S) = \{r \in R | \exists \ell \in S, (\ell, r) \in E\}$ be the neighbors of S. Then G has a matching of size $|L|$ if and only if $|N(S)| \geq |S|$ for all $S \subseteq L$

   (c) Use Konig's or Halls theorems to show that in a bipartite graph where every vertex has the same degree d (also called a d-regular bipartite graph), there is a perfect matching.

   (d) Use the above part to show that the edges of a d-regular bipartite graph can be colored using d colors so that all edges incident on each vertex have different colors.

2. **Some Matching Reductions.** Suppose you have an algorithm that solves max-weight perfect matchings (MaxWPM) for all non-negative weight functions. Give reductions that allow you to solve (a) min-weight perfect matchings (MinWPM) and (b) min-weight max cardinality matchings (MinWMaxM)i.e., among all matchings of size equal to $MM(G)$, find the one with least weight. If the MinWPM and MinWMaxM instances are bipartite, ensure that the reductions give you MaxWPM instances that are bipartite too.

3. **Min-weight Perfect Matchings.** Suppose $G = (L, R, E)$ is a undirected bipartite graph with (possibly negative) integer edge weights w_e, suppose M is some perfect matching. Let H_M be the digraph obtained by directing edges in $E \setminus M$ from left-to-right and putting weights w_e on them, and directing all edges in M from right-to-left and putting weights $-w_e$ on them.

   (a) Show that H_M has a negative-weight cycle if and only if M is not a min-weight perfect matching.

   (b) Consider the algorithm: Start with any perfect matching M. While there exists a negative-weight cycle C in H_M, set $M \leftarrow M \Delta C$. Show that you will eventually get a MwPM.
   If $T(n)$ is the time to find a negative-weight cycle in an n-node graph, and $W := max_{e \in E} |w_e|$, a naive bound on the runtime of the above algorithm would be $O(nW) \cdot T(n)$, plus the time to find the first perfect matching.

   (c) For matching M, denote $w(M) := \sum_{e \in M} w_e$. Show that if $w(M) > w(M)$ then there exists a cycle in H_M with weight-ratio1 at most $\frac{w(M) - w(M)}{n}$, where M^* is a min-weight perfect matching.

   (d) Suppose we change the algorithm in (b) to use the minimum weight-ratio cycle C at each step. Bound the runtime of this algorithm by $O(n \log(nW)) \cdot T'(n)$, plus the time to find the first perfect matching. Here $T'(n)$ is the time to find a minimum weight-ratio cycle.

4. **A Degree of Smoothness..** In the Max-Cut problem, we are given a graph $G = (V, E)$ with edge weights w_e, and want to find a subset $S \subseteq V$ such that the weight of the edges crossing the cut (denoted as

$\partial S := \{(u,v) \in E : u \in S, v \in V \setminus S\}$) is maximized. In the smoothed analysis setting, the edge weights are random variables W_e supported on $[0,1]$, whose density function is bounded by $1/\sigma$, as in lecture. Suppose the maximum degree in G is Δ.

One algorithm for max-cut is the simple local-search algorithm: start with any subset S. Now if there is a vertex v which can be moved from S to $V \setminus S$ (or vice versa) to improve the weight $w(\delta S)$, make the change. Show that the expected number of improving steps in such an algorithm is bounded by $O(n^3 2^\Delta/\sigma)$. (Hint: recall the argument we used for the $2OPT$ local search heuristic for TSP.)

5. **Blocking Experts (due to Avrim Blum).** Here is a variation on the deterministic WeightedMajority algorithm, designed to make it more adaptive.

   - Each expert begins with weight 1 (as before).
   - We predict the result of a weighted-majority vote of the experts (as before).
   - If an expert makes a mistake, we penalize it by dividing its weight by 2, but only if its weight was at least $\frac{1}{4}$ of the average weight of experts.

   Prove that in any contiguous block of trials (e.g., the 51st day through the 77th day), the number of mistakes made by the algorithm is at most $O(m + \log N)$, where m is the number of mistakes made by the best expert *in that block*, and N is the total number of experts.

6. **Expertise in Experts..** Here are some problems to help you internalize the experts proof.

   (a) Show that if we change the basic weighted majority algorithm to reduce the weight by (1ϵ) instead of $1/2$, we get

   $$\text{our total loss} \leq 2(1+\epsilon) \cdot (\text{total loss of expert } i) + O(\frac{\log N}{\epsilon}).$$

   (b) Show that if the loss vectors are in $[-\infty, 1]^N$, then our analysis of Hedge gives us that for any expert $i \in [N]$,

   $$\sum_t \langle \vec{p}^{(t)}, \vec{\ell}^{(t)} \rangle \leq \sum_t \ell_i^{(t)} + \frac{\ln N}{\epsilon} + \epsilon \sum_t \langle \vec{p}^{(t)}, (\vec{\ell}^{(t)})^2 \rangle$$

   Here $\langle u, v \rangle$ is the usual inner product between two vectors u, v, and $(\vec{u})^2$ denotes a vector whose jth entry is u_j^2.

   (c) Use the previous part to show that:

   i. If losses are in $[0,1]^N$ and $\epsilon \leq \frac{1}{2}$, then infer that our loss is

   $$\sum_t \langle \vec{p}^{(t)}, \vec{\ell}^{(t)} \rangle \leq \frac{1}{1-\epsilon} (\sum_t \ell_i^{(t)} + \frac{\ln N}{\epsilon}) \leq (1+2\epsilon) \sum_t \ell_i^{(t)} + \frac{2 \ln N}{\epsilon}.$$

   This gives a nice multiplicative-additive guarantee. (Hint: $x^2 \leq x$ for $x \in [0,1]$).

   ii. Similarly, if we have only gains (i.e., if losses are all in $[1,0]^N$), then we get

   $$textourgain \geq (1-2\epsilon)(\text{ gain of best expert }) - \frac{2 \ln N}{\epsilon}.$$

   (d) Suppose the loss functions $\ell^{(t)} \in [-\rho, \rho]^N$, considver the vectors $p^{(t)}$ obtaine by running Hedge with "learning rate" $\epsilon' = \frac{\epsilon}{2\rho}$ where we feed it the "fake" loss function $\tilde{\ell}^{(t)} - \vec{\ell}^t/\rho \in [-1,1]^N$. Show they satisfy

   $$\sum_t \langle \vec{p}^{(t)}, \vec{\ell}^{(t)} \rangle \leq \sum_t \ell_i^{(t)} + \frac{2\rho^2 \ln N}{\epsilon} + \epsilon T/2$$

   Hence, show that the average regret is at most ϵ if $T \geq \frac{2\rho^2 \ln N}{\epsilon^2}$.

   (e) Suppose we change the update step in the Hedge algorithm to be $w_j^{(t+1)} \leftarrow w_j^{(t)} \cdot (1 - \epsilon \vec{\ell}_t^{(j)})$. Alter the analysis to show that the regret after T steps is still $O\left(\frac{\log N}{\epsilon} + \epsilon T\right)$.

Problems Answer problems #1 and #2, and any two out of the remaining three problems.

1. **Matchings and Matroids.** Let $\mathcal{M} = (U, \mathcal{I})$ be a matroid. For $X \in \mathcal{I}$, define a bipartite graph $H_{\mathcal{M}}(X) = (X, U \setminus X, E)$, where $(x, y) \in E$ iff $x \in X, y \in U \setminus X$ and $(X + y - x) \in \mathcal{I}$. (that is, we can exchange x for y and get another independent set of the same size).

   (a) Let $X, Y \in \mathcal{I}$ be two independent sets with $|X| = |Y|$ Show that $H_{\mathcal{M}}(X)$ contains a perfect matching on $X \Delta Y$– it contains a matchin $M \subseteq E$ which matches exactly the elements of $X \setminus Y$ to those of $Y \setminus X$ (Hint: Hall's theorem from the exercises).

   (b) (A partial converse.) For $X \in \mathcal{I}$, suppose there exists a set $Y \subseteq U$ such that $h - |m(X)$ contains a *unique* perfect matching on $X \Delta Y$. Show that $Y \in \mathcal{I}$. HInt: one proof is via induction on $|Y \setminus X|$. You may use part (a).

2. **Boost your score.** Suppose you have a set of points $S \subseteq \mathbb{R}^d$, and for each point $x \in S$ you are also given a weight $w(x) \geq 1$ and a label $\ell(x) \in \{0, 1\}$. A function $f : \mathbb{R}^d \to \{0, 1\}$ is called a hypothesis. If $W := \sum_{x \in S} w(x)$, the error of the hypothesis on the set S is

$$\mathrm{err}_{S,l,w}(f) := \frac{1}{W} \sum_{x \in S} w(\mathbf{x}) \cdot 1_{(f(x) \neq \ell(x))},$$

   i.e., the fraction of weight on points in S that f labels incorrectly.

   Suppose you know an algorithm $\mathcal{A}$ that is weak learner for class G: given any set $S \subseteq \mathbb{R}^d$ (along with its weights $w()$ and the labels $\ell()$), this weak learner will return a function $f \in G$, such that $\mathrm{err}_{S,\ell,w}(f) \leq \frac{1}{2} - \eta$ for some $\eta > 0$. If η is small, this means the returned hypothesis is only slightly better than a random coin toss.

   Give an algorithm that, given the point set S, along with the weights $w()$ of the points and their labeling $\ell()$, runs $\mathcal{A}$ for $T = O(\frac{1}{\eta^2} \log \frac{1}{\epsilon})$, times and outputs a function $\hat{f}$ of the form

$$\hat{f}(x) = \textbf{majority } (f_1(x), f_2(x), \ldots, f_T(x))$$

   for $f_i \in G$, such that $\mathrm{err}_{S,\ell,w}(\hat{f}) \leq \epsilon$. (Hint: Start off with the simple case where the weights of the points in S are $w(x) = 1$ for all $x \in S$; it contains all the ideas. Also, recall the analysis of the Hedge or Weighted Majority even if you dont use these algorithms themselves.)

3. **A Market-Based Bipartite Max-Matching Algorithm.** Give $G = (I, B, E)$, the left vertices are items, the right are buyers. Let $n = \max(|I|, |B|)$. For each edge $(i, b) \in E$, let $v_{ib} = 1$; if $(i, b) \notin E$, then $v_{ib} = 0$. The intial prices are $p_i = 0$; define the utility of buyer b for item i under prices $\overline{p} = (p_1, p_2 \ldots, p_{|I|})$ to be

$$u_{ib}(\overline{p}) = max\{v_{ib} - p_i, 0\}.$$

   For some parameter $\delta \in (0, 1)$, consider the following algorithm:

   - A1 Start with the empty matching $M = \emptyset$.
   - A2 Pick any unmatched buyer b such that its highest-utility item i has $u_{ib}(\overline{p}) \geq \delta$. Match (i, b), which ma require dropping (i, b') for some other b' from the current matching. After this operation, i is assigned to b rather than b'. Raise the price $p_i \leftarrow p_i + \delta$.
   - A3 If for each unmatched buyer, $\max_i u_{ib}(\overline{p}) < \delta$, let the current matching be denoted by M_1. Use the augmenting paths algorithm to augment M_1 to a maximum matching M^*.

     We now show that M_1 is a "large" matching, and so step $(A3)$ does not take "much" time.

   (a) Show that $|M_1| \geq |M^*| - n\delta$. Hint: you could try to use the following LP:

$$\begin{array}{ll} \max \sum_{ib} v_{ib} x_{ib} & \\ \sum_b x_{ib} \leq 1 & \forall i \in I \\ \sum_i x_{ib} \leq 1 & \forall b \in B \end{array}$$

   and show that $|M_1| \geq OPT_{LP} - n\delta$ using the duals.

(b) Show how to use the appropriate data structures to implement the algorithm so that all executions of step (A2) can be done in a tota of $O(\frac{1}{\delta} \cdot (m + n \ln n))$ time.

(c) Choose the value of δ so thta the running time of the entire algorithm is as close to $O(m\sqrt{n})$ as possible. You may assume that each augmenting path can be found in $O(m)$ time and hence (A3) takes a total of $O(m) \times (|M^*| - |M_1|)$ time.

Observe that this algorithm is different from the one in class: there the prices went up in lock-step, here we just pick a single item and unilaterally raise its price by a tiny constant. We also dont get a perfect matching here at the end, we have to do some correction (Step (A3)). BTW, what can you say about the performance of such an algorithm when you have general weights?

4. **Sherman-Morrison and Dynamic Algorithms.** . Suppose graph G_0 has a perfect matching to start off, and someone starts adding and removing edges. Each update step adds or removes a single edge e_i from G_{i1} to give G_i. At each time i, we want to answer "does G_i still have a perfect matching? Of course, we could simply answer the question for each G_i independently, but let us try to be smarter.

The Sherman-Morrison(-Woodbury) formula[1] says: for any non-singular $n \times n$ matrix A, and any n-dimensional vectors u, v,

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^TA^{-1}u.}$$

(Assuming that $1 + v^TA^{-1}u \neq 0$). Moreover, the matrix determinant lemma says

$$\det(A + uv^T) = \det(A)(1 + v^TA^{-1}u)$$

These tell us how the inverse and determinant of a matrix change under rank-one updates, as long as these updates dont make the matrices singular.

(a) Use these to show that for $i \geq 1$, if $G_0, G_1 \ldots, G_{i-1}$ all have perfect matchings, we can answer the question for G_i correctly with probability $1O(i/poly(n))$, but in time $\tilde{O}(n^2)$. (Be sure to mention what field $\mathbb{F}_q$ you are working over.)[2] You may assume that the graphs G_i are all bipartite.

(As soon as G_{i-1} has no perfect matching, your algorithm may be incorrect for all subsequent $G_j, j \geq i$. Bonus: suggest ways to handle this shortcoming. Also bonus: suggest ways to reduce remove the dependence on i in the above probability bound.)

This general idea of using Sherman-Morrison for rank-one updates gives dynamic algorithms for several other problems. Here is another example, where we need to count the number of paths between vertices in DAGs.

(b) For a square matrix M such that $M^k = 0$ for some positive integer k, show that

$$(I - M)^{-1} = \sum_{i=0}^{k-1} M^i.$$

(c) For a DAG G, let A be the $ntimesn$ adjacency matrix: $A_{ij} = 1$ if and only if (i, j) is an arc in G. Show that the (i, j) th entry of $(IA)^{-1}$ is the number of paths between vertices i, j in G. Hence, infer that you can compute the number of paths from i to j, for all i, j, in time $O(n^\omega)$.

(d) Show how to maintain these path counts under edge additions and deletions to the graph, as long as it remains a DAG. (Make sure you argue that you dont perform any operations that cause matrices to become singular, etc.) The time for each update should be $O(n^2)$.

5. **Isolated? No Man is an Island.** For this problem we assume that $G = (U \cup V, E)$ is a bipartite graph with a perfect matching and a weight on the edges $w : E \to \mathbb{R}_{\geq 0}$. We say that w is isolating if the minimum-weight perfect matching in G is unique.

You may use the following lemma:

[1]See this stackexchange discussion for intuition, or the wikipedia article for other pointers.

[2]The naive way would be to use the Tutte-Lovasz approach for each timestep independently, which would require $O(n^\omega) \gg O(n^2)$ time per update: your goal is to do better

Lemma 1. *If the edge weights are picked independently from a set of size* $\geq 2|E| = 2m$*, the resulting weight function is isolating with probability* $\geq \frac{1}{2}$.

This requires $m \cdot \lceil \log_2 2m \rceil$ bits of information. In this exercise, we construct an isolating weight with $O(\log^2 n)$ bits of randomness.

(a) (Do not submit) Given cycle $C = \langle u_1, v_1, u_2, v_2, ..., u_k, v_k, u_1 \rangle$ in G, define its imbalance:

$$imb_w(C) := w_{u_1 v_1} - w_{v_1 u_2} + w_{u_2 v_2} - w_{v_2 u_3} \cdots + w_{u_k v_k} - w_{v_k u_1}|.$$

Show that if w is a weifht satisfying $imb_w(C) \neq 0$ for all cycles C, the w is isolating.

(b) Let h be a graph with n nodes and no cycles of length $\leq r$ and let $r' = 4 \cdot \lfloor r/2 \rfloor$. Then show that H has at most n^4 cycles of length $\leq r'$.

(c) Let C be a cycle in G such that $imb_w(G) \neq 0$. Let E_1 be the union of all minimum-weight perfect matchings in G. Show then that $G_1 = (U \cup V, E_1)$ does not contain a cycle in C.

Hint: Consider the perfect matching polytope and let x denote the point which is the average of all the min-weight perfect matchings in G. Show that if $imb_w(C) \neq 0$ then you can construct a new point that is cheaper than x.

(d) Consider a graph G with n nodes and some $s \geq 1$. You may assume that there is an oracle O which uses $\log(ns)$ random bits to produce a non-negative weight w such that for any set of s cycles $\{C_1, C_2, \ldots, C_s\}$,

$$\mathbb{P}[imb_w(C_i) \neq 0 \text{ for all } 1 \leq i \leq s] \geq 1 - \frac{1}{n}.$$

Use the above oracle O along with parts (b) and (c), to construct an isolating weight w using $O(\log^2 n)$ bits of randomness with probability at least $1 - \frac{O(\log n)}{n}$.