Problem Set 1 CS265/CME309, Autumn 2023
Due: Friday 10/6, 11:59pm on Gradescope

Please follow the homework policies on the course website.

---

1. **(5 pt.) [Perfect Matchings.]**

   Let $G = (V, E)$ be a *bipartite graph* with n vertices on each side. A *perfect matching* in G is a list of edges $M \subset E$ so that every vertex in V is incident to exactly one edge.

   For example, here is a bipartite graph G (on the left), and a perfect matching in G (shown in **bold** on the right):

   Your goal is to determine if the graph G has a perfect matching.

   There are efficient deterministic algorithms for this problem, but in this problem you'll work out a simple randomized one.[1]

   (a) **(2 pt.)** Recall that the *determinant* of an $n \times n$ matrix A is given by

$$\det(A) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} A_{i,\sigma(i)},$$

   where the sum is over all permutations $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$, and where $\operatorname{sgn}(\sigma)$ denotes the signature[2] of the permutation σ. (For example, if $n = 3$, then the function $\sigma : \{1, 2, 3\} \to \{1, 2, 3\}$ that maps $1 \mapsto 2$, $2 \mapsto 1$, $3 \mapsto 3$ is a permutation in S_3. The signature of σ happens to be -1, although as noted in the footnote, if you haven't seen this definition before, don't worry about it).

   Let A be the $n \times n$ matrix so that

$$A_{ij} = \begin{cases} x_{ij} & (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$$

   where the x_{ij} are variables, and $(i, j) \in E$ if and only if the i-th vertex on the left and the j-th vertex on the right are connected by an edge in G. Notice that $\det(A)$ is a multivariate polynomial in the variables x_{ij}.

   Explain why $\det(A)$ is not identically zero if and only if G has a perfect matching.

[1] This randomized algorithm has the advantage that (a) it generalizes to all graphs (not necessarily bipartite), and (b) it can be parallelized easily. Moreover, it's possible to generalize it to actually recover the perfect matching (and not just decide if there is one or not).

[2] The *signature* of a permutation is defined as -1 if the permutation can be written as an odd number of transpositions, and $+1$ otherwise. **The exact definition isn't important** to this problem, all you need to know is that it's either ± 1 in a way that depends on σ.

(b) **(3 pt.)** Use the part above to develop a randomized algorithm for deciding whether or not there is a perfect matching. Your algorithm should run in $O(n^3)$ operations. If G has no perfect matching, your algorithm should return "There is no perfect matching" with probability 1. If G has a perfect matching, your algorithm should return "There is a perfect matching" with probability at least 0.9.

You should clearly state your algorithm and explain why it has the desired properties.

[**HINT:** *You may use the fact that one can compute the determinant of a matrix* $A \in \mathbb{R}^{n \times n}$ *in* $O(n^3)$ *operations.*]

(c) **(0 pt.) [Optional: this won't be graded.]** Extend your algorithm to actually *return* a perfect matching. And/or, extend your algorithm to non-bipartite graphs. As a hint, consider the matrix

$$A = \begin{cases} x_{ij} & \{i,j\} \in E \text{ and } i < j \\ -x_{ji} & \{i,j\} \in E \text{ and } i \geq j \\ 0 & \text{else} \end{cases}$$

2. **(8 pt.)** Suppose you are rolling a fair, 6-sided die repeatedly.

(a) **(4 pt.)** What is the expected number of rolls until you get two 3's in a row (counting both 3's)? Justify your answer.

[**HINT:** *The answer is not* 36*...*]

[**HINT:** *If you find yourself doing a tedious computation, try to think of a simpler way. Perhaps look to the mini-lecture on linearity of expectation for some inspiration...*]

(b) **(4 pt.)** What is the expected number of rolls until you get a 3 followed by *either* a 3 or a 4 (counting both rolls)? Justify your answer.

(c) **(0 pt.) [Optional: this won't be graded]** What is the expected number of rolls until you get a 3 followed by a 4 (counting both rolls)?

3. **(10 pt.)** Suppose you are given a fair coin (that is, it lands heads/tails with probability 1/2 each) and want to use it to "simulate" a coin that lands heads with probability exactly 1/3. Specifically, you will design an algorithm whose only access to randomness is by flipping the fair coin (repeatedly, if desired), and your algorithm should return "heads" with probability exactly 1/3 and "tails" with probability exactly 2/3.

(a) **(4 pt.)** Prove that it is *impossible* to do this if the algorithm is only allowed to flip the fair coin at most $1,000,000,000$ times.

[**HINT:** *Read the next two parts of the problem first...*]

(b) **(4 pt.)** Design an algorithm for the above task that flips the fair coin a finite number of times *in expectation.*

(c) **(2 pt.)** Show that for *any* value v in the interval $[0, 1]$, there is an algorithm that flips a fair coin at most 2 times in expectation, and outputs "heads" with probability v and "tails" with probability $1 - v$.

Note: if you do this part correctly, you can write "follows from (c)" in part (b) and get full credit for both parts.

[**HINT:** *Think about representing the desired probability in its binary representation.*]

4. **(8 pt.)** Its that time of year when folks all around campus are deciding whether or not to purchase a parking permit. The tricky part is that you don't know how many times a parking attendant will check your car over the course of the year—maybe they will just check each day for the first week of the quarter to scare you into buying a permit, maybe it will be every day, or maybe there are no parking attendants. Don't fret–we're here to help you navigate this big decision. [The simple answer might be to just not have a car...]

   Suppose each parking ticket is \$1 and a parking permit costs \$10 and you can purchase a parking permit at any point (i.e. after paying 3 tickets, you can decide to buy a permit). If a parking attendant checks your car and you don't have a permit, you will get a ticket. Let T represent the total number of times that a parking attendant will check your car. If we knew T, the optimal strategy would be to buy a parking permit at the very beginning of the quarter if $T > 10$, and otherwise, just pay the parking tickets, and we would incur a cost of $min(T, 10)$. A *strategy* is a policy for deciding when we buy a permit—namely how many tickets are we willing to receive before we buy a permit. Given a strategy, S, define our *regret* to be a function of T corresponding to the amount we pay using strategy S if the attendant came T times, minus the cost of the optimal policy if we had known in advance what T was:

$$Regret_S(T) := Cost(S, T) - min(T, 10).$$

   . Our goal will be to find a strategy S that minimizes the *worst-case* regret, $\max_T(Regret_S(T))$.

   (a) **(2 pt.)** Show that the worst-case regret of any deterministic strategy is at least 10. Namely, a deterministic strategy corresponds to buying a permit after exactly w tickets for some fixed value of $w \in \{0, 1, \ldots, \infty\}$. Show that for each such w, there exists a T that would cause us to incur a regret of at least 10.

   (b) **(2 pt.)** We will now consider randomized strategies: such strategies can be thought of as a distribution D_S over $\{0, 1, 2, \ldots\}$: we draw w from this distribution, and will buy a permit after the wth ticket. For any T, each such a strategy will incur an *expected* cost $E[Cost(S, T)] = \sum_{i=0}^{\infty} \Pr[w = i] \cdot Cost(w, T)$, where $Cost(w, T)$ is T if $w > T$ [ie we never bought the permit and paid T tickets], and $10 + w$ if $w \leq T$ [ie we paid the first w tickets then bought the \$10 permit]. Suppose our goal is to minimize the *worst-case expected regret*:

$$\max_T \left(E[Cost(S, T)] - min(T, 10) \right).$$

   Is minimizing this quantity a reasonable goal? Discuss in two or three sentences. (If you think the worst-case nature of this quantity is too pessimistic/paranoid/hedged, try to propose an alternate metric.)

   (c) **(3 pt.)** Suppose our randomized strategy corresponds to drawing w from a geometric distribution with parameter p. Namely, before any tickets could be given, and after each ticket received, we flip a coin that lands heads with probability p, and if the coin does land heads, we buy a permit. What is the choice of p that minimizes our worst-case expected regret, *and what is the worst-case expected regret* for that choice of p? *[Justify your answer, though feel free to write down a messy expression for the worst-case expected regret as a function of p, and optimize by writing a python script. Do make sure that your answer sanity-checks—-p shouldn't be too big or too small, and the worst-case expected regret should be quite a bit better than the* 10 *we get with a deterministic strategy.]*

(d) **(1 pt.)** Does the policy from the previous part seem like something you might actually use in real-life (assuming that permits cost 10X the cost of a ticket)? Discuss in at most two sentences.

(e) (Bonus + 1 point) Suppose a ticket costs \$K but a permit costs \$Z dollars. Intuitively, the optimal choice of p should scale according to K/Z. Suppose $p = c\frac{K}{Z}$ for some constant c. Find the optimal constant c in the limiting case as K/Z goes to zero. In the limit, by what factor is this worst-case expected regret better than the Z worst-case regret of deterministic strategies?

(f) (Bonus/food-for-thought: 0 points) Either in the case of a permit costing \$10, or in the limiting case as a permit cost gets large, what is the optimal randomized strategy? Namely, if we can pick the time at which we get a permit, w, according to *any* distribution—not necessarily a geometric distribution—what distribution should we use, and how much better is the worst-case expected regret versus that of the best geometric distribution from parts (c) or (d)?