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

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

SOLUTION:

(a) Fix a permutation σ, and consider the matching (not necessarily a matching in G) given by matching i to $\sigma(i)$. If this is a matching in G (that is, if $(i, \sigma(i)) \in E$ for all i), then

$$\prod_{i=1}^{n} A_{i,\sigma(i)} = \prod_{i=1}^{n} x_{i,\sigma(i)} =: \mathbf{x}_\sigma$$

is a monomial that is not identically zero. On the other hand, if this is not a matching in G, then

$$\prod_{i=1}^{n} A_{i,\sigma(i)} = 0.$$

Thus, we can write the determinant as

$$\det(A) = \sum_{\sigma \in S_n} \text{sgn}(\sigma)\mathbf{1}[\sigma \text{ corresponds to a perfect matching }]\mathbf{x}_\sigma.$$

In particular, if there is no perfect matching, then the sum is empty, and $\det(A) \equiv 0$. On the other hand, if there is a perfect matching, then there is at least one monomial in the sum. Since $\mathbf{x}_\sigma \neq \mathbf{x}_{\sigma'}$ for $\sigma \neq \sigma'$, if there are multiple nonzero monomials they cannot cancel, and so $\det(A) \not\equiv 0$.

(b) The algorithm is:

- Choose a set $S \subseteq \mathbb{R}$ of size $10n$, arbitrarily.
- Construct the matrix A', where x_{ij} variables in A are replaced by corresponding values $r_{ij} \in S$ is chosen uniformly at random for each $\{i, j\} \in E$. This takes time $O(n^2)$.
- Compute $\det(A')$ in $O(n^3)$ operations.
- If $\det(A') = 0$, return "no perfect matching." Otherwise, return "perfect matching."

To see that this is correct, notice that if G has no perfect matching, then by part (a), $\det(A)$ will be identically equal to zero, so the algorithm above will always return "no perfect matching." On the other hand, if there is a perfect matching, then $\det(A)$ will not be an identically zero polynomial. In this case the Schwartz-Zippel Lemma implies that the probability that $\det(A') = 0$ is at most $n/|S|$, since the degree of $\det(A)$ is n. Our choice of $|S| = 10n$ means that we will succeed with probability at least $9/10$, as desired.

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)?

SOLUTION:

(a) The answer is 42. Let X be the expected number of rolls until you get two 3's. We have

$$\mathbb{E}X = \frac{2}{36} + \frac{5}{36}(2 + \mathbb{E}X) + \frac{5}{6}(1 + \mathbb{E}X).$$

This is because:

- With probability $1/36$, we get 3,3 on the first two rolls, and the answer is 2.
- With probability $(1/6) \cdot (5/6) = 5/36$ we get 3 and then not-a-3 on the first two rolls. In this case the answer is 2 plus however many times it takes to roll two 3s in the future.
- With probability $5/6$, we get a non-3 on the first roll. In this case the answer is 1 plus however many times it takes to roll two 3s in the future.

Solving for $\mathbb{E}X$, we get $\mathbb{E}X = 42$.

(b) Following similar logic above, the answer is 21. Let X be the expected number of rolls until you get two 3's or a 3,4. We have

$$\mathbb{E}X = \frac{2}{36} \cdot 2 + \frac{4}{36}(2 + \mathbb{E}X) + \frac{5}{6}(1 + \mathbb{E}X).$$

Solving for $\mathbb{E}X$ yields 21.

(c) The same logic is difficult (for me) to extend to this last part, since if we first roll "33" then there's not a nice way to get $\mathbb{E}X$ to show up directly so that we can solve for it. Instead we have to introduce another variable, Y, which is the number of rolls left after we've just rolled a 3. Then we have

$$\mathbb{E}X = 1 + \frac{5}{6}\mathbb{E}X + \frac{1}{5}\mathbb{E}Y,$$

since if we haven't just rolled a 3, we need to roll once, and then if we've rolled something other than a 3 (with probability 5/6), we start again with $\mathbb{E}X$. On the other hand, if we did roll a 3 (with probability 1/6), then we have to wait $\mathbb{E}Y$ in expectation. Similarly, we can write

$$\mathbb{E}Y = 1 + \frac{4}{6}\mathbb{E}X + \frac{1}{6}\mathbb{E}Y,$$

because we have to make one roll; if it's a 4 then that's it, we're done. If it's a 3, then we again have $\mathbb{E}Y$ to wait in expectation. And if it's neither a 3 nor a 4 (with probability 4/6), then we're back at the very beginning and have to wait $\mathbb{E}X$ again in expectation. Solving these two equations for $\mathbb{E}X$ and $\mathbb{E}Y$ yields $\mathbb{E}X = 36$, $\mathbb{E}Y = 30$, so the answer is 36.

[**Note 1:** this sort of argument will be a lot easier once we have some notation from Markov chains, which we will see later in the course.]

[**Note 2:** It may seem confusing that the answer to 2(a) is larger than the answer to 2(c) – what accounts for this difference? if we have to get two 3s in a row, then when we get a single 3 followed by a value that is not a 3, we get reset all the way back to our starting state. On the other hand, if we are trying to get a 3 followed by a 4, then after we get a single 3, we have two possible favorable outcomes for the next roll: a 4 makes us win, and a 3 keeps us in a more favorable "one step from victory" state rather than booting us back to the beginning.]

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.*]

SOLUTION:

(a) For any event X defined in terms of the outcomes of at most $n = 1,000,000,000$ fair coin flips, the probability of X will be some integer multiple of $1/2^n$. [If this isn't clear to you, stop and think about it!] Since 2^n contains no multiples of 3, there is no integer k for which $k/2^n = 1/3$, and hence no algorithm flipping at most n coins can output "heads" with probability exactly 1/3.

(b) There are many correct solutions. One nice algorithm is the following: flip the fair coin twice. If the outcomes are HH, return "heads", if the outcomes are HT or TH return "tails". If the outcome is TT then repeat. The expected number of flips of this algorithm is $2 + 2\frac{1}{4} + 2\frac{1}{4^2} + 2\frac{1}{4^3} \ldots = 2\frac{4}{3} < 3$. To see why the algorithm is correct, note that the probability of returning "heads" immediately after the first two tosses is $\frac{1}{4}$, the probability of returning heads after flipping exactly 4 four coins is $1/4^2$ (namely the first two coins must land TT and the next two must land HH), and in general, the probability of returning "heads" immediately after flipping $2i$ coins is $1/4^i$. Hence the overall probability of returning "heads" is $1/4 + 1/4^2 + 1/4^3 + \ldots = 1/3$, as desired.

(c) The algorithm is the following. Write $v = \sum_{i=1}^{\infty} b_i 2^{-i}$, where $b_i \in \{0, 1\}$. Then consider the following protocol:

- For $i = 1, 2, \ldots$:
  - Flip a fair coin.
  - If $b_i = 1$ and the coin is heads, return HEADS and stop flipping.
  - if $b_i = 0$ and the coin is tails, return TAILS and stop flipping.

First, we analyze the probability that this procedure outputs HEADS. We have

$$\Pr[\text{alg outputs HEADS}] = \sum_{i=1}^{\infty} \Pr[\text{alg outputs HEADS on iteration } i].$$

The probability that the algorithm outputs HEADS on iteration i is

$$\begin{aligned} \Pr[\text{output HEADS on } i] &= \Pr[\text{output HEADS on } i | \text{still flipping at time } i] \cdot \Pr[\text{still flipping at time } i] \\ &= \begin{cases} 0 \cdot 2^{-(i-1)} & b_i = 0 \\ \frac{1}{2} \cdot 2^{-(i-1)} & b_i = 1 \end{cases} \\ &= b_i 2^{-i}. \end{aligned}$$

Thus,

$$\Pr[\text{alg outputs HEADS}] = \sum_{i=1}^{\infty} b_i 2^{-i} = v.$$

Next we analyze the expected number of flips. At each flip of the algorithm (assuming we make it that far), we stop flipping with probability 1/2. Thus, by the analysis that we have seen in class for the expectation of a geometric random variable, the expected number of flips until we stop is 2. (Note that if the binary representation of v is not infinitely long, then we could stop the algorithm above early and output TAILS; in this case the expected number of flips might be less).

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?

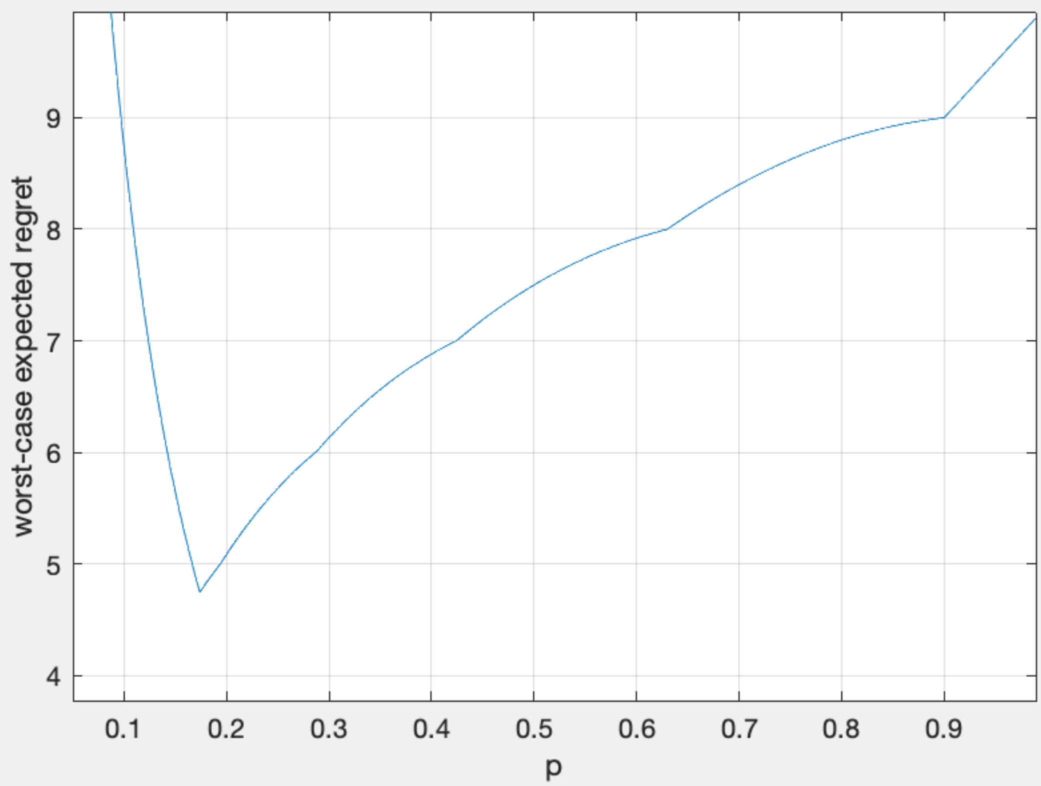
(f) (Bonus/food-for-thought: 0 points) Either in the case of a permit costing \$10, or in the limiting case as a permit cost gets large, what is the optimal randomized strategy? Namely, if we can pick the time at which we get a permit, w, according to *any* distribution—not necessarily a geometric distribution—what 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)?

SOLUTION:

(a) Given a deterministic strategy that would purchase a permit after the rth ticket, if r is infinite (ie we never purchase a permit), then for any $T \geq 20$ we would have regret at least 10. For finite r, in the case that $T = r$, we would have regret exactly 10.

(b) Any thoughtful response received full credit.

(c) The easiest solution is to write a script that computes the expected regret for a given value of p and T, then for each value of p, finding the worst-case T, and then finding the p that minimizes that. Here is a plot of the worst-case expected regret as a function of p. Note that for $p \approx 0$, the regret is unbounded (ie you might need to pay 100 tickets!), and for $p \approx 1$, the regret is close to 10, since you'll probably buy the permit on day 0, but if $T = 0$ you've wasted the cost of the permit and should have just not bought the permit. The minimum is at $p \approx 0.175$, in which case the worst-case expected regret is roughly 4.75.



(d) Any thoughtful response received full credit.

(e) Solutions to [most] bonus questions won't be posted, but happy to discuss in office hours.

(f) Solutions to [most] bonus questions won't be posted, but happy to discuss in office hours.