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) In one direction, suppose G has a perfect matching. Then there exist at least one permutation σ such that $\forall i \in \{1, ..., n\}$, the product $\prod_i A_{i,\sigma(i)}$ is comprised of all variables. Consequently, if those variables take none-zero values, the product will be non-zero and the determinant will not be identically zero.

In the other direction, suppose the determinant is not identically zero. Then at least one term in the sum is non-zero. For any non-zero term in the sum, its product $\prod_i A_{i,\sigma(i)}$ is non-zero, and therefore, all the variables in the product must be non-zero. Since each row and column of A can only appear once in each product, the variables in this product are a perfect matching for G.

(b) The determinant of A is some polynomial of the variables $\{X_{i,j}\}$. From Lecture 1, we know from the Schwartz-Zippel Polynomial Identity Test (Lecture 1, Theorem 3.1) tests whether a polynomial is identically zero in the following manner: if the polynomial is identically zero, the algorithm will always output "P=0" (or in our case, $det(A) = 0$), and if the polynomial is not, the algorithm will output "P$\neq$ 0" with probability $1 - degree(P)/|S|$.

In our problem, $degree(P) = degree(det(A)) = n$. We wish to choose S to control the correctness probability:

$$0.9 = 1 - n/|S| \Rightarrow |S| = 10n$$

We need to fix a set S of possible variable values with size 10 times n to ensure that the Schwartz-Zippel Polynomial Identity Test, applied to the polynomial $det(A)$, returns the correct answer (with appropriate probability) if $det(A)$ is not identically zero.

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) Let X be the integer number of flips to get two consecutive 3s. A tree is a good way to think this through. Suppose we initialize $X \leftarrow 0$. With probability 5/6, we roll not a 3 and thus increment $X \leftarrow X + 1$. On the other branch, with probability 1/6, we roll a 3; conditional on being on this branch, we have a $\frac{1}{6}$ probability of rolling a 2nd 3 and a $\frac{5}{6}$ probability of rolling not a 3, which would force us to restart and thus add 2 to our total:

$$X = \frac{5}{6}(X+1) + \frac{1}{6}\Big(\frac{1}{6}2 + \frac{5}{6}(X+2)\Big)$$

Simplifying, we have:

$$X = \frac{35}{36}X + \frac{42}{36}$$

Using linearity of expectation, the expectation is:

$$\mathbb{E}X = \frac{35}{36}\mathbb{E}X + \frac{42}{36}$$

Solving for the expected value:

$$\mathbb{E}X = 42$$

(b) Similar branching logic as above can work. Let X be the integer number of flips to get either a 3 followed by a 3 or a 3 followed by a 4. Initialize $X \leftarrow 0$. With probability 5/6, we roll not a 3 and thus increment $X \leftarrow X + 1$. On the other branch, with probability 1/6, we roll a 3; conditional on being on this branch, we have a $\frac{2}{6}$ probability of rolling a 3 or a 4, and a $\frac{4}{6}$ probability of rolling not a 3 or 4, which would force us to restart and thus add 2 to our total:

$$X = \frac{5}{6}(X+1) + \frac{1}{6}\Big(\frac{2}{6}2 + \frac{4}{6}(X+2)\Big)$$

Simplifying, we have:

$$X = \frac{17}{18}X + \frac{7}{6}$$

Using linearity of expectation, the expectation is:

$$\mathbb{E}X = \frac{17}{18}\mathbb{E}X + \frac{7}{6}$$

Solving for the expected value:

$$\mathbb{E}X = 21$$

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) get full credit for both parts.

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

SOLUTION:

(a) Read my answer to (c), then return here. Since $v = 1/3$ has no finite binary representation (it is $0.\overline{01}$), in order to accurately represent $v = 1/3$, an unbounded number of flips is necessary. No finite number of flips will allow for achieving a true probability of heads $= 1/3$.

(b) Follows from (c).

(c) The high-level idea is similar to binary search (in computer science) or a stick breaking process (in stochastic processes). Fix $v \in [0, 1]$ e.g., $0.\bar{6}$. For our algorithm, it's helpful to pretend we have a stick painted $0.\bar{6}$ red (for heads) and $0.\bar{3}$ blue (for tails). The algorithm is: iteratively, break the stick in half, then flip a coin. If the coin comes up tails, select the left half and throw the right half away; if the coin comes up heads, select the right half and throw the left half away. If the selected half of the stick has a single

color, return that color (recall: red for heads, blue for tails); otherwise, if the selected half of the stick has both colors, repeat/recurse.

Why will this work? Because the probability of needing to recurse to a particular depth will fall $\propto 2^{-d}$ where d is the depth. It also requires the ability to recurse infinitely if one wishes to correctly model v, thus explaining why no finite number of flips can exactly model $v = 1/3$, since there is no finite binary representation of $1/3$.

Formally, let X be the number of coin flips (alternatively, recursion depth). With probability $1/2$, we're done. With probability $1/2$, we (may) need to repeat/recurse.

$$X \leq \frac{1}{2}(1) + \frac{1}{2}(1 + X)$$

Consequently,

$$\mathbb{E}X \leq 1 + \frac{1}{2}\mathbb{E}X$$

Solving:

$$\mathbb{E}X \leq 2$$

This is guaranteed to output heads with probability v.

4. **(8 pt.)** You are walking down a street full of restaurants, trying to decide where to eat for lunch. There are n restaurants on the street, but you haven't been to this street before, so you're not sure what's coming up, and you don't want to walk all the way to the end of the street and then potentially double back to pick the best-looking restaurant.

   Instead, your plan is to walk for a little while to get a sense of the restaurant distribution, then eat at the first place that looks good after that. Formally, you will walk past the first m restaurants, and assign them all a rating based on how good they look. Then, starting with the $m + 1$'st restaurant you pass, you will eat at the first one you find that you rate higher than *all* of the first m restaurants. (Suppose for simplicity that there are no ties in the ratings). If you get to the end of the street and don't find such a restaurant, you don't get lunch :(.

   Suppose that the restaurants are in a uniformly random order on the street.

   (a) **(4 pt.)** Let E be the event that you end up eating at the best-looking (i.e., your highest-rated) restaurant on the whole street. Show that

   $$\mathbb{P}[E] = \frac{m}{n} \sum_{j=m+1}^{n} \frac{1}{j-1}.$$

   [**HINT:** *Let E_i be the event that the i'th restaurant you pass is the best-looking one, and that you eat there. What is the probability of E_i?*]

(b) **(2 pt.)** Bound $\sum_{j=m+1}^{n} \frac{1}{j-1}$ to show that

$$\frac{m}{n}(\ln n - \ln m) \leq \mathbb{P}[E] \leq \frac{m}{n}(\ln(n-1) - \ln(m-1))$$

[**HINT:** *Compare the sum to an integral.*]

(c) **(2 pt.)** Explain how to pick m so that

$$\mathbb{P}[E] \geq 1/e - o(1).$$

(Here, the $o(1)$ term means something that tends to 0 as $m, n \to \infty$.)
[Note: the $o(1)$ term was added 10/5/22.]
[**HINT:** *What m should you pick to maximize $\frac{m}{n}(\ln n - \ln m)$?*]

SOLUTION:

(a) First, consider a single E_i.

$$\begin{aligned}\mathbb{P}[E_i] &= \mathbb{P}(\text{ith restaurant is best})\mathbb{P}(i > m)\mathbb{P}(\text{eat at ith}|\ \text{i > m \& ith is best}) \\ &= \frac{1}{n}\mathbb{I}\{i > m\}\frac{m}{i-1}\end{aligned}$$

where $m/(i-1)$ is the answer to the question: what is the probability that the max of $i-1$ items occurs in the first m slots. The full probability is then:

$$\mathbb{P}[E] = \sum_{j=1}^{n} \mathbb{P}[E_j] = \sum_{j=1}^{n} \frac{1}{n}\mathbb{I}\{j > m\}\frac{m}{j-1} = \frac{m}{n}\sum_{j=m+1}^{n}\frac{1}{j-1}$$

where we rely on the events being disjoint (since the best restaurant can't be at multiple indices).

(b) For the upper bound:

$$\sum_{j=m+1}^{n}\frac{1}{j-1} \leq \sum_{j=m}^{n-1}\frac{1}{j-1} \approx \int_{m}^{n}\frac{1}{j-1}dj = \log(n-1) - \log(m-1)$$

For the lower bound:

$$\sum_{j=m+1}^{n}\frac{1}{j-1} \geq \int_{m+1}^{n+1}\frac{1}{j-1}dj = \log(n) - \log(m)$$

We can then bound via:

$$\log(n) - \log(m) \leq \log(n-1) - \log(m) \leq \log(n-1) - \log(m-1)$$

Then multiply by $\frac{m}{n} > 0$ to get the desired result:

$$\frac{m}{n}(\log(n) - \log(m)) \leq \mathbb{P}[E] \leq \frac{m}{n}(\log(n-1) - \log(m-1))$$

(c) We know that $\frac{m}{n}(\log(n) - \log(m)) \leq \mathbb{P}[E]$. To maximize $\mathbb{P}[E]$, we can instead maximize the lower bound $\frac{m}{n}(\log n - \log m)$. To do so, we take the derivative and set equal to 0:

$$0 = \partial_m \Big(\frac{m}{n}(\log n - \log m) \Big)$$

Rearranging:

$$0 = \frac{\log n}{n} - \frac{1}{n} \log m - \frac{m}{n} \frac{1}{m}$$

Solving for m:

$$m = \exp(\log(n) - 1) = \frac{n}{e}$$

Substituting back in:

$$\mathbb{P}[E] \geq \frac{m}{n}(\log(n) - \log(m)) = \frac{1}{e}(\log n - \log \frac{n}{e}) = \frac{1}{e}$$