Note: Your TA probably will not cover all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. They are deliberately made long so they can serve as a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

Philosophy of analyzing randomized algorithms. The first step is to always identify a bad event—you want to identify when your randomness makes your algorithm fail. We will review some techniques from class using the following problem as our “test bed”.

Let \( G \) be a bipartite graph with \( n \) left vertices, and \( n \) right vertices on \( n^2 - n + 1 \) edges.

- Prove that \( G \) always has a perfect matching.
- Give a polynomial in \( n \) time algorithm to find this perfect matching.

We will now use some common techniques to analyze the following algorithm BlindMatching:

- Let \( \pi \) and \( \sigma \) be independent and uniformly random permutations of \([n]\).
- If \( \{\pi(1), \sigma(1)\}, \{\pi(2), \sigma(2)\}, \ldots, \{\pi(n), \sigma(n)\} \) is a valid matching output it.
- Else output failed.

Union Bound. Suppose \( X_1, \ldots, X_n \) are (not necessarily independent) Bernoulli random variables (i.e. random variables valued \{0, 1\}). Then, we have the following identity:

\[
\Pr[X_1 + \cdots + X_n \geq 1] \leq \Pr[X_1 = 1] + \Pr[X_2 = 1] + \cdots + \Pr[X_n = 1].
\]

Now, using union bound, we analyze our algorithm for BlindMatching. Note that an output

\[
M = (\{\pi(1), \sigma(1)\}, \ldots, \{\pi(n), \sigma(n)\})
\]

is a valid perfect matching exactly when all edges of the form \( \{\pi(i), \sigma(i)\} \) are present in \( G \). A “bad event” happens if any of those pairs are not edges in \( G \).

Let \( X_i \) be the indicator of the event that \( \{\pi(i), \sigma(i)\} \) is not present in our graph.

1. What is the probability that \( X_i = 1 \)?

2. Use the union bound to upper bound the probability that \( M \) is not a valid perfect matching.

3. Conclude that \( G \) has a valid perfect matching.

The upper bound obtained on the probability of our bad event, i.e. of \( M \) not being a valid perfect matching, is fairly high. In light of this, we introduce the technique of amplification.
Amplification. The philosophy of amplification is that if we have a randomized algorithm that fails with probability $p$, we can repeat the algorithm many times and aggregate the output of all the runs to produce a new output such that the failure probability of the randomized algorithm is significantly smaller. Now consider the following algorithm \texttt{SpamBlindMatching}:

- Run \texttt{BlindMatching} independently $T$ times.
- If at least one of the runs outputted a valid perfect matching, return the output of such a run.
- Else output failed.

To see the effectiveness of amplification, let us answer the following questions:

1. What is tight upper bound on the failure probability of \texttt{SpamBlindMatching}?

2. How large should we set $T$ if we want a failure probability of $\delta$?

Notice that the failure probability of \texttt{SpamBlindMatching} is not only lower than that of \texttt{BlindMatching}, but it can also be adjusted for based on the number of “amplifications” we make!

Now we switch gears and turn our attention to concentration phenomena and its usefulness in analyzing randomized algorithms.

Markov’s inequality. Let $X$ be a nonnegative valued random variable, then for every $t \geq 0$:

$$\Pr[|X| \geq t \mathbb{E}[X]] \leq \frac{1}{t}. \quad (1)$$

1. Markov’s inequality is \textit{false} for random variables that can take on negative values! Give an example.

2. Give a tight example for Markov’s inequality. In particular, given $\mu$ and $t$, construct a random variable $X$ such that $\mu = \mathbb{E}[X]$ and $\Pr[|X| \geq t \mu] = \frac{1}{t}$.

Chebyshev’s inequality. Let $X$ be any random variable with well-defined variance\footnote{In this course, all random variables will have well-defined variance (i.e. $\text{Var}[] < \infty$).}, then

$$\Pr \left[ |X - \mathbb{E}[X]| > t \sqrt{\text{Var}[X]} \right] \leq \frac{1}{t^2}. \quad (2)$$

To see the above inequality in action, consider the following problem:

Let $B$ be a bag with $n$ balls, $k$ of which are red and $n-k$ of which are blue. We do not have knowledge of $k$ and wish to estimate $k$ from $\ell$ independent samples (with replacement) drawn from $B$.

Let $X$ be the number of red balls sampled.
1. What is $E[X]$?

2. What is $\text{Var}[X]$?

3. Choose a value for $\ell$ and give an algorithm that takes in $n$ and $X$ and outputs a number $\tilde{k}$ such that $\tilde{k} \in [k - \varepsilon \sqrt{k}, k + \varepsilon \sqrt{k}]$ with probability at least $1 - \delta$. 
1 Traveling Salesman Problem

In the lecture, we learned an approximation algorithm for the Traveling Salesman Problem based on computing an MST and a depth first traversal. Suppose we run this approximation algorithm on the following graph:

![Graph Image]

The algorithm will return different tours based on the choices it makes during its depth first traversal.

1. Which DFS traversal leads to the best possible output tour?

2. Which DFS traversal leads to the worst possible output tour?

3. What is the approximation ratio given by the algorithm in the worst case for the above instance? Why is it worse than 2? (Hint: Consider the triangle inequality on the graph).

2 Maximum Coverage

In the maximum coverage problem, we have $m$ subsets of the set $\{1, 2, \ldots, n\}$, denoted $S_1, S_2, \ldots, S_m$. We are given an integer $k$, and we want to choose $k$ sets whose union is as large as possible.

Give an efficient algorithm that finds $k$ sets whose union has size at least $(1 - 1/e) \cdot OPT$, where $OPT$ is the maximum number of elements in the union of any $k$ sets. In other words, $OPT = \max_{i_1, i_2, \ldots, i_k} | \bigcup_{j=1}^{k} S_{i_j} |$. Just the algorithm description and justification for the lower bound on the number of elements your solution contains is needed.

(Recall the set cover algorithm from lecture, and use that $(1 - 1/n)^n \leq 1/e$ for all integers $n$)
3 Second smallest global mincut

Recall that a *cut* of an undirected, unweighted graph \((V, E)\) is a partition of the vertices into two non-empty sets \(S\) and \(V \setminus S\). The *weight* of such a cut is the number of edges crossing the cut, which we denote \(|E(S, V \setminus S)|\). Karger’s contraction algorithm from class finds a cut of minimum weight with probability \(1 - p\), with runtime \(O(poly(n) \cdot \log(1/p))\), where \(n := |V|\) and \(m := |E|\).

In this problem, we will be concerned with finding not the smallest, but rather the second smallest minimum cut. Ties are broken arbitrarily. For example, consider the graph below:

![Graph](image)

It has two cuts of weight one, given by \(S = \{0\}\) or \(S = \{3\}\). All other cuts have weight at least two. Since we break ties arbitrarily, we would say in this example both the smallest and second smallest cuts have weight equal to 1.

(a) Give a general formula for the number of cuts in a graph with \(n\) vertices and \(m\) edges. **Note:** we are asking for the number of cuts, not necessarily minimum cuts.

(b) Focus on a particular second smallest mincut \(S^*\). Consider running just the first step of Karger’s contraction algorithm, where we contract a uniformly random edge. Write down a value \(\alpha\) such that the probability that the contracted edge crosses \((S^*, V \setminus S^*)\) is guaranteed to be at most \(\alpha\). Your answer should depend on the weight \(k\) of this second smallest mincut, as well as \(n\), but not on \(m\). Also, you should give such an \(\alpha\) that is as small as possible to get full credit (e.g., writing \(\alpha = 1\), though correct, will receive no credit).

(c) Starting with \(n\) vertices, imagine doing \(n - 3\) random contraction steps in sequence. How many “supervertices” are left?

(d) Suppose your answer in part (c) is \(c\). Imagine then picking a uniformly random cut (not necessarily a mincut) on this graph of \(c\) supervertices to get a resulting cut of the original graph. Show that the probability that this resulting cut is \(S^*\) is \(\Omega(1/n^2)\).

(e) Give a \(O(poly(n) \cdot \log(1/p))\) time algorithm for finding the second smallest mincut in a graph with success probability at least \(1 - p\). You need not calculate the precise exponent of \(n\) in the \(poly(n)\) term; any polynomial dependence suffices. *(Hint: \((1 + x)^T \leq e^{(xT)}\) for all \(x \in \mathbb{R}\) and \(T \geq 0\).)*