You can collaborate with your classmates, but be sure to list your collaborators with your answer. If you get help from a published source (book, paper etc.), cite that. The answer must be written by you and you should not be looking at any other source while writing it. Also, limit your answers to one page, preferably less —you just need to give enough detail to convince the grader.

Typeset your answer in latex. If you don’t know latex please write by hand clearly and scan it into pdf using your smartphone or the scanner in the mail room.

§1 Prove that Karger’s algorithm does not work for finding the minimum s-t cut in unweighted, undirected graphs. That is, design an unweighted, undirected graph $G$, with two nodes $s$, $t$, such that repeatedly contracting a random edge that does not contract $s$ and $t$ to the same supernode outputs a minimum s-t cut with probability $2^{-\Omega(n)}$.

Hint: try to prove that the algorithm works, and see which step fails. Use this to guide your example.

§2 In class we saw a hash to estimate the size of a set. Change it to estimate frequencies. Thus there is a stream of packets each containing a key and you wish to maintain a data structure which allows us to give an estimate at the end of the number of times each key appeared in the stream. The size of the data structure should not depend upon the number of distinct keys in the stream but can depend upon the success probability, approximation error etc. Just shoot for the following kind of approximation: if $a_k$ is the true number of times that key $k$ appeared in the stream then your estimate should be $a_k \pm \epsilon \left( \sum_k a_k \right)$. In other words, the estimate is going to be accurate only for keys that appear frequently ("heavy hitters") in the stream. (This is useful in detecting anomalies or malicious attacks.) Hint: Think in terms of maintaining $m_1 \times m_2$ counts using as many independent hash functions, where each key updates $m_2$ of them.

§3 Show that given $n$ numbers in $[0,1]$ it is impossible to estimate the value of the median within say 1.1 factor with $o(n)$ samples. (Hint: to show an impossibility result you show two different sets of $n$ numbers that have very different medians but which generate —whp—identical samples of size $o(n)$.)

Now calculate the sample size needed (as a function of $t, n, \delta$) so that the following is true: with probability $1 - \delta$, the median of the sample has at least $n/2 - t$ numbers less than it and at least $n/2 - t$ numbers more than it.

§4 A cut is said to be a $B$-approximate min cut if the number of edges in it is at most $B$ times that of the minimum cut. Show that a graph has at most $(2n)^{2B}$ cuts that are $B$-approximate. (Hint: Run Karger’s algorithm until it has $2B + 1$ supernodes.
What is the chance that a particular $B$-approximate cut is still available? How many possible cuts does this collapsed graph have?)

§5 Consider an unweighted, undirected graph $G = (V, E)$ whose mincut has value $c = \omega(\log n)$. You would like to create a sparser (weighted) graph $G' = (V, E')$ with $E' \subseteq E$, but such that the weight of each cut is approximately preserved. Specifically: sample every edge $e \in E$ with probability $p$. If $e$ is sampled, add $e$ to $E'$ with weight $1/p$. You want that for all $S \subseteq V$, $\text{CUT}_{G'}(S) \in (1 \pm \varepsilon) \cdot \text{CUT}_G(S)$. The smaller you make $p$, the sparser $G'$ will be (and therefore easier to store/operate/etc.). The bigger you make $p$ the more likely you are to actually preserve the value of all cuts (consider e.g. $p = 1$). As a function of $\varepsilon$, what is a sufficiently large $p$ to guarantee with probability $1 - \delta$ that for all $S \subseteq V$, $\text{CUT}_{G'}(S) \in (1 \pm \varepsilon) \cdot \text{CUT}_G(S)$? (Treat $\delta, \varepsilon$ as constants w.r.t $n$. For instance, you may assume that $f(n) > g(\delta, \varepsilon)$ for any $f$ satisfying $\lim_{n \to \infty} f(n) = \infty$, and any $g(\cdot, \cdot)$).

Hint: The previous question might help! You will also need to use the fact that $c$ is sufficiently large in your proof.

§6 Consider the following problem: there are $n > k$ independent (but not identically distributed) non-negative random variables $X_1, \ldots, X_n$ drawn according to distributions $D_1, \ldots, D_n$. Initially, you know each $D_i$ but none of the $X_i$s.

Starting from $i = 1$, each $X_i$ is revealed one at a time. Immediately after it is revealed, you must decide whether to “accept $i$” or “reject $i$,” before seeing the next $X_{i+1}$. You may accept at most $k$ elements in total (that is, once you’ve accepted $k$ times, you must reject everything that comes after). Your reward at the end is $\sum_{i \mid i \text{ was accepted}} X_i$.

- For general $k$, design a policy that guarantees expected reward at least $(1 - O(\sqrt{\ln(k)/k})) \cdot \mathbb{E}_{X_1, \ldots, X_n \sim D_1, \ldots, D_n} \left[ \sum_{j=1}^k X_r(j) \right]$, where $r$ is a permutation from $[n]$ to $[n]$ satisfying $X_{r(1)} \geq X_{r(2)} \geq \ldots \geq X_{r(n)}$ (i.e. the policy gets expected reward at least $(1 - O(\sqrt{\ln(k)/k}))$ times the expected sum of top $k$ weights, which is the best you could do even if you knew all the weights up front).

Hint: Try to set up a simple policy that can be analyzed using a Chernoff bound.

- Come up with an example showing that it is not possible to improve the above guarantee beyond $(1 - \Omega(1/\sqrt{k}))$ (which is optimal - you do not need to prove this).

Hint: an example exists whose complete proof should fit in half a page.

§7 (extra credit) The chromatic number of a graph is the smallest number of colors required to color a graph where no two adjacent vertices have the same color. Show that the chromatic number of $G(n, 1/2)$ is about $n/(2 \log n)$ with high probability ($G(n, 1/2)$ is a graph on $n$ nodes where the edge between $i$ and $j$ is present with probability $1/2$, independently for all pairs).