princeton university fall '19 cos 521:Advanced Algorithms

# Homework 1

Out: *Sep 15* Due: *Sep 30*

**Instructions:**

- Upload your solutions (to the non-extra-credit) as *a single* PDF file (one PDF total) to Mechanical TA. Please anonymize your submission (do not list your name in the PDF title or in the document itself). If you forget, it's OK.
- If you choose to do extra credit, upload your solution to the extra credits as a single PDF file to Mechanical TA. Please again anonymize your submission.
- You may collaborate with any classmates, textbooks, the Internet, etc. Please attach a brief "collaboration statement" listing any collaborators at the end of your PDF (if you forget, it's OK). You should write up your solutions individually.
- For each problem, you should aim to keep your writeup below one page. For some problems, this may be infeasible, and for some problems you may write significantly less than a page. This is not a hard constraint, but part of the assignment is figuring out how to easily convince the grader of correctness, and to do so concisely. "One page" is just a guideline: if your solution is longer because you chose to use figures (or large margins, display math, etc.) that's fine.
- Each problem is worth twenty points (even those with multiple subparts).

**Problems:**

§1 In class, we saw that, when hashing $m$ items into a hash table of size $O(m^2)$, the expected number of collisions was $< 1$. In particular, this meant we could easily find a "perfect" hash function of the table that has no collisions.

Consider the following alternative scheme: build two tables, each of size $O(m^{1.5})$ and choose a separate hash function for each table independently. To insert an item, hash it to one bucket in each table and place it in the emptier bucket (tie-break randomly).

(a) Show that, if we're hashing $m$ items, with probability 1/2, there will be no collisions in either table. You may assume a fully random hash function.

(b) Modify the above scheme to use $O(\log m)$ tables, each of size $O(m)$. Prove that this approach yields a collision-free hashing scheme with space $O(m \log m)$. Again, you may assume a fully random hash function.

§2 Prove that (the natural variant of) 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 no parallel edges), 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 $2^{-\Omega(n)}$.[1]

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

§3 Let $X = \{x_1, \dots, x_n\}$ be $n$ (not necessarily distinct) numbers in $[0, 1]$. You do not know the numbers, but may repeatedly *sample* a uniformly random element of $X$.

(a) Prove that no algorithm using $o(n)$ samples can estimate the value of the median within a factor of 1.1. That is, any algorithm which (possibly randomly) maps $m = o(n)$ samples to a guess at the median is off by a factor of at least 1.1 with probability at least 1/3.

Hint: come up with two instances with very different medians, but which look the same after $o(n)$ samples with high probability.

(b) Instead, say we seek a number $y$ such that at least $n/2-t$ elements of $X$ exceed $y$, and $n/2 - t$ numbers are less than $y$. Prove that if we take $m = O(n^2 \ln(1/\delta)/t^2)$ samples, and let $y$ denote the median of the $m$ samples, then $y$ has this property with probability at least $1 - \delta$.

§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 all undirected graphs have 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 $E' \subset E$, 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 $S \subseteq V$, $\mathrm{CUT}_{G'}(S) \in (1 \pm \varepsilon) \cdot \mathrm{CUT}_G(S)$. 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. $p = 1$).

Prove that for all constant $\varepsilon$, that for any graph $G$ on $n$ nodes with mincut $c = \omega(\log n)$, taking $p = O(\frac{\ln n}{\varepsilon^2 c})$ results with probability at least $1-o(1)$ in a $G'$ satisfying: for all $S \subseteq V$, $\mathrm{CUT}_{G'}(S) \in (1 \pm \varepsilon) \cdot \mathrm{CUT}_G(S)$.

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 random process: there are $n + 1$ coupons $\{0, \dots, n\}$. Each step, you draw a uniformly random coupon independently with replacement, and you

[1] To be clear: the algorithm is guaranteed to output *a* s-t cut, it just might not be the minimum.

repeat this until you have drawn *all coupons in* $\{1, \ldots, n\}$ (that is, you may terminate without ever drawing 0). Prove that, with probability at least $1 - O(1/n)$, you draw the 0 coupon at most $O(\log n)$ times.

Hint: You may find it easier to use Chernoff bounds and union bounds than previous analysis you've seen related to the coupon collector problem.

**Extra Credit:**

§1 (extra credit) 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 $\sum_{i|i \text{ was accepted}} X_i$.

(a) For general $k$, design a policy that guarantees expected reward at least $(1 - O(\sqrt{\frac{\ln(k)}{k}})) \cdot \mathbb{E}_{X_1,\ldots,X_n \leftarrow D_1,\ldots,D_n}[\sum_{j=1}^{k} X_{r(j)}]$, 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 $(1 - O(\sqrt{\frac{\ln(k)}{k}}))$ 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.

(b) 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. You may use without proof the fact that if $X$ is the number of coin flips which land heads from $k$ independent fair coin flips, then $\mathbb{E}[|X - k/2|] = \Theta(\sqrt{k})$.

§2 (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 $(1 \pm o(1)) \cdot n/(2 \log n)$ with probability $1 - o(1)$ ($G(n, 1/2)$ is a graph on $n$ nodes where the edge between $i$ and $j$ is present with probability $1/2$, independently for all pairs).