PRINCETON UNIVERSITY FALL '24 COS 521:ADVANCED ALGORITHMS

Homework 1

Out: *Sep 13* Due: *Sep 29*

Instructions:

- Upload your solutions (to the non-extra-credit) as *a single* PDF file (one PDF total) to Gradescope. 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 Gradescope. Please again anonymize your submission.
- You may discuss ideas for solutions with any classmates, textbooks, the Internet, etc. Please attach a brief "collaboration statement" listing any collaborators at the end of your PDF. **You must 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 insert it only in the emptier bucket (tie-break lexicographically).

(a) Show that, if we're hashing m items, with probability 1/2, there will be no collisions in either table (a collision occurs when multiple distinct elements are inserted into the same bucket in the same 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 again that with probability 1/2, there will be no collisions in any table. 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. Specifically, 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 In this problem, we investigate whether an algorithm can compute the median of a given set of numbers when it can only access the input set via independent samples.

(a) Let A be an algorithm the input to which is a collection of m independent samples drawn uniformly at random from an arbitrary set $X = \{x_1, \ldots, x_n\}$. Prove that if $m = o(n)$ then A must fail compute the median of X within a multiplicative error of 1.1 with probability at least 1/3. 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. To prove that your algorithm must behave similarly on these instances, recall that your algorithm's behavior is completely determined by the samples it sees (you can avoid tedious calculations by cleverly using this observation).

(b) Say we relax the goal and ask for the algorithm above to output 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 the following process for matching n jobs to n processors. In each step, every job picks a processor at random. The jobs that have no contention on the processors they picked get executed, and all the other jobs back off and then try again. Jobs only take one round of time to execute, so in every round all the processors are available. Show that all the jobs finish executing after $O(loglogn)$ steps, with high probability.

§6 Consider the following random process: there are $n + 1$ coupons $\{0, \ldots, n\}$. Each step, you draw a uniformly random coupon independently with replacement, and you 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.

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

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_is.

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).