Princeton Univ Fall '25	COS 521:Advanced Algorithms
Homework 1	
Out: *Sep 8*	Due: *Sep 21*

Instructions:

- Upload your non-extra solutions to Gradescope in a single PDF file, and mark your solution to each problem. Please make sure you are uploading the correct PDF! Please anonymize your submission (i.e., do not list your name in the PDF), but if you forget, it's OK.
- If you choose to do extra credit, upload your solution to the extra credits as a single separate PDF file to Gradescope. Please again anonymize your submission.
- You may collaborate with any classmates, textbooks, the Internet, etc. Please upload a brief "collaboration statement" listing any collaborators as a separate PDF on Gradescope (if you forget, it's OK). But always **write up your solutions individually**.
- For each problem, you should have a solid writeup that clearly states key, concrete lemmas towards your full solution (and then you should prove those lemmas). A reader should be able to read any definitions, plus your lemma statements, and quickly conclude from these that your outline is correct. This is the most important part of your writeup, and the precise statements of your lemmas should tie together in a correct logical chain.
- A reader should also be able to verify the proof of each lemma statement in your outline, although it is OK to skip proofs that are clear without justification (and it is OK to skip tedious calculations). Expect to learn throughout the semester what typically counts as 'clear'.
- You can use the style of Lecture Notes and Staff Solutions as a guide. These tend to break down proofs into roughly the same style of concrete lemmas you are expected to do on homeworks. However, they also tend to prove each lemma in slightly more detail than is necessary on PSets (for example, they give proofs of some small claims/observations that would be OK to state without proof on a PSet).
- Each problem is worth twenty points (even those with multiple subparts), unless explicitly stated otherwise.

Problems:

§1 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.

§2 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 for integers $B \geq 1$.

Hint: Run Karger's algorithm until it has $2B$ supernodes. What is the chance that a particular B-approximate cut is still available? How many possible cuts does this collapsed graph have?

§3 (a) Consider the following random process: there are n coupons $\{1, \ldots, n\}$. Each step, you draw a uniformly random coupon independently with replacement, and you repeat this until you have drawn *all coupons in* $\{1, \ldots, n\}$. Prove that with probability at least $1 - 1/n$, the process takes $O(n \log n)$ steps.

(b) 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 w.h.p. after $O(\log \log n)$ steps.

Hint: Try to argue that if there are currently ϵn unmatched jobs, then in the next round roughly $\epsilon^2 n$ jobs remain unmatched.

§4 Suppose that there are n items whose sizes $X_1, X_2, \ldots, X_n$ are drawn independently from Unif$[0, 1]$. In the (NP-Hard) bin packing problem, the goal is to find the *minimum* number $f(X_1, X_2, \ldots, X_n) \in \mathbb{N}$ of bins such that it's possible to pack these items into f unit-sized bins (i.e., each item is assigned to a bin and the sum of sizes of items assigned to each bin is at most 1).

(i) Show that $\mu := \mathbb{E}[f]$ is $\Omega(n)$.

(ii) Show that for any $i \in \{1, \ldots, n\}$, if the size of i-th item changes then the value of f changes by at most 1. That is, prove that

$$|f(X_1, \ldots, X_i \ldots, X_n) - f(X_1, \ldots, X_i', \ldots X_n)| \leq 1,$$

where each $X_1, X_2, \ldots X_i, X_i', X_{i+1} \ldots, X_n$ is chosen *arbitrarily* in $[0, 1]$.

Remark: The above condition (ii) on f is known as Bounded-Difference condition. McDiarmid's concentration inequality is that if f satisfies (ii) then $\Pr_{X_1,\ldots,X_n}[f >$

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

$(1+\epsilon)\mu] \leq \exp\big(-\frac{\epsilon^2\mu}{10n}\big)$. Thus, even though computing f is NP-Hard for adversarial inputs, for random inputs it's w.h.p. only $\pm O(\sqrt{n})$ around its mean.

§5 *Linear probing* is another popular way to resolve hash collisions. Imagine that we are inserting n items to a hash table of size $m \geq n$ using a *uniformly random hash function* h that maps the items to the m slots. The items are inserted one at a time, and assume there are no deletions. To insert the next item x into the hash table A, we first check if $A[h(x)]$ is already occupied, if not, we store x in $A[h(x)]$. Otherwise, we check if $A[h(x)+1]$ is occupied, if not, we store x in $A[h(x)+1]$, etc. Item x is stored in the first empty slot $A[h(x)+i]$ for some $i \geq 0$. Let us further assume the algorithm is able to find an empty slot before reaching $h(x)+i > m$.

Suppose $m = 2n$, and we have inserted $n-1$ items in this way. Consider the process of inserting the last item x. Show that for any sufficiently large R, the probability that x is stored in $A[h(x)+R]$ is at most $\exp(-\Omega(R))$.

Hint: For some $L \geq 0$, bound the probability that $A[h(x)-L], \ldots, A[h(x)+R-1]$ are all occupied, and $A[h(x)-L-1]$ is empty. What is the possible range of hash values of the items stored in these slots?

Extra Credit:

§1 Show that for linear probing, when $m = (1+\epsilon)n$, the expected time to insert the last item is $\Theta(\epsilon^{-2})$.