Problem Set 2

Randomized Algorithms

Due Tuesday, October 3

1. [Karger.] Consider a sequence of n unbiased coin flips, and look at the length of the longest contiguous sequence of heads.

   (a) Show that you are unlikely to see a sequence of length $c + \log_2 n$ for $c > 1$ (give a decreasing bound as a function of c).

   (b) Show that with high probability you will see a sequence of length $\log_2 n - O(\log_2 \log_2 n)$. Note: this observation can be used to detect cheating. When told to fake a random sequence of coin tosses, most humans will avoid creating runs of this length under the mistaken assumption that they dont look random.

2. Negative Association.

   (a) Let $X_1, \ldots, X_n$ be independent but not necessarily identically distributed random variables. Let $\sigma_1, \ldots, \sigma_n$ be drawn from a permutation distribution on $[n]$. Are the variables $Y_i = X_{\sigma_i}$ negatively associated?

   (b) Recall the following algorithm from class for estimating the mean of an unknown random variable X with mean μ and variance σ^2. Given $n = mB$ samples $x_1, \ldots, x_n$, choose $m = O(\log(1/\delta))$ blocks of size $O(1/\epsilon^2)$. Output

   $$\widehat{\mu} := \operatorname*{median}_{i \in [m]} \operatorname*{mean}_{j \in [B]} x_{(B-1)i+j}.$$

   We showed that the result is within $\epsilon\sigma$ of μ with probability $1 - \delta$. Now, suppose that our sample $x_1, \ldots, x_n$ were not independent, but negatively associated. Would the same result hold?

3. [Karger.] In class we proved that the two-choices approach improves the maximum load to $O(\log \log n)$. A generalization is that choosing the least loaded of d choices reduces the maximum load to $O(\log_d \log n)$. Explain what changes to the proof are needed to derive this result. Give only the diffs; do not bother writing a complete proof.

4. [Karger.] In class, we showed that cuckoo hashing achieves worst case constant time lookups and expected constant time insertion/deletion, with $O(n)$ space to store n items. Show how to get the same guarantees, but using only $(1 + \epsilon)n$ space for a small constant ϵ. For this problem, assume that you have access to a perfectly random hash function. **Hint:** Use the following ideas:

   - Probing more than twice in a table increases the chances of finding an empty cell.
   - If after some probes you fail to find an empty cell, move the failed item into an "overflow" table that uses cuckoo hashing.