


Homework 8 Due Thursday, October 31, 2024

Page limit You can submit **at most** 2 pages per problem, even if the problem has multiple parts. If you submit a longer solution for some problem, only the first sheet of paper will be graded.

Reminder Collaboration is permitted, but you must write the solutions *by yourself without assistance*, and be ready to explain them orally to the instructor if asked. You must also identify your collaborators and whether you gave help, received help, or worked something out together. Getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden.

Exercises Please practice on exercises in Chapter 5 of Mitzenmacher-Upfal.

Problems

1. **(Bucket Sort)** Exercise 5.9 from MU.
2. **(Poisson Pumpkin Picking )** Linus Van Pelt is in a pumpkin patch searching for The Great Pumpkin at midnight on Halloween. He decides to pass the time by carving Jack-O-Lanterns, and picks pumpkins uniformly and independently at random. Suppose that a p_1 fraction of the pumpkins are too small, a p_2 fraction of the pumpkins are too big, and a p_3 fraction of the pumpkins are “just right”, where $p_1 + p_2 + p_3 = 1$. Linus picks a Poisson number of pumpkins with mean μ . Let X, Y , and Z be the number of pumpkins that are too small, too big, and “just right”, respectively. Prove that X, Y , and Z are *independent* Poisson random variables with means $p_1\mu$, $p_2\mu$, and $p_3\mu$, respectively.

For fun (no extra points) name the comic strip and fairytale involved in this problem.

3. **(Poisson Fish)** You and your roommate bike to Walden Pond. Your friend proposes an algorithm \mathcal{A} that takes descriptions of fish sampled uniformly at random from the pond and outputs an estimate of the number of species in the pond. Sampling is with replacement, of course; it is state law, and who knows what else is swimming in that pond. The difficulty is that algorithm \mathcal{A} needs the number of samples it gets as input to be distributed according to Poisson distribution. Specifically, it draws $N \sim \text{Poisson}(s)$, that is, from the Poisson distribution with parameter s , and then takes N samples. However, to fish at Walden pond, you had to get a permit, and the permit says that you can catch (and then release, of course) at most $2s$ fish.

You think to yourself that effectively you will be running algorithm \mathcal{A}' instead of algorithm \mathcal{A} . Algorithm \mathcal{A}' runs algorithm \mathcal{A} as a black box if the number of samples \mathcal{A} requests is at most $2s$; otherwise algorithm \mathcal{A}' returns “fail”. Your goal is to prove that \mathcal{A} and \mathcal{A}' have essentially the same behavior. Let random variable Y be the estimate returned by \mathcal{A} and random variable Y' be the estimate returned by \mathcal{A}' .

- (a) Use Chebyshev’s inequality to prove the following **claim**: For every set S of possible outputs,

$$|\Pr[Y \in S] - \Pr[Y' \in S]| \leq \frac{2}{s}.$$

Your roommate (who is a complexity theorist) says that you just proved that algorithms with a fixed-size sample can simulate Poisson algorithms. He defined *Poisson algorithms* as algorithms that take N uniform samples, where N is drawn from a Poisson distribution. He is wondering if Poisson algorithms can simulate algorithms with a fixed-size sample. Now suppose you have an algorithm \mathcal{A} that takes s samples. Your goal is to design a Poisson algorithm \mathcal{A}' so that both algorithm satisfy the claim above.

- (b) State your algorithm \mathcal{A}' .
- (c) Prove the **claim**. Apply Chernoff-type bounds for Poisson distribution (instead of Chebyshev's inequality).

Note: Both Chernoff bounds and Chebyshev's inequality are applicable for both parts (a) and (c).

- 4*. **(Optional, no collaboration¹, based on the bonus part of the Noisy Binary Search Problem on the midterm)** You are given an integer x that is stored in a sorted array A of n distinct integers, $a_1 < a_2 < \dots < a_n$. You would like to find out the index $i \in [n]$ such that $x = a_i$. You can only access A via queries of the form $\text{LEQ}(i)$, where you specify $i \in [n]$, and you get back a YES/NO answer to the question "Is $x \leq a_i$?". The difficulty is that the answer is computed by a randomized algorithm, which is correct with probability at least 0.9. The answers to all queries are independent (even when the same query is asked repeatedly).

This problem is called *Noisy Binary Search*, and your goal is to develop an algorithm that makes $O(\log n)$ queries² and solves this problem with high probability.

- (a) **(Part(a) from the miterm, don't hand in)** Consider the function $\text{RobustBisect}(\ell, r)$ that takes two indices $\ell, r \in [n]$, such that $\ell < r$, and returns one of the three possible outputs, specified below. Let $i = \lceil \frac{\ell+r}{2} \rceil$. The outputs are
 - LOW if $x \in (a_\ell, a_i]$;
 - HIGH if $x \in (a_i, a_r]$;
 - OUT-OF-RANGE if $x \notin (a_\ell, a_r]$.

Give a randomized algorithm \mathcal{B} that solves RobustBisect with probability at least 0.8 by making LEQ queries.

- (b) Use \mathcal{B} as a subroutine in an algorithm \mathcal{A} based on the binary search in order to determine i such that $x = a_i$. Design a strategy such that once \mathcal{B} has been correct at least $\log n$ more times than it has been incorrect (in other words, the number of runs that give the correct answer minus the number of runs that give the incorrect answer is at least $\log n$), algorithm \mathcal{A} outputs i such that $x = a_i$ exactly and correctly³.
- (c) ⁴Use Chernoff-Hoeffding bounds to prove that one can solve the Noisy Binary Search problem with $O(\log n)$ queries LEQ with probability at least $1 - \frac{1}{n}$.

¹If you discussed the bonus problem after the exam, it is OK; just work on this problem on your own from now on.

²In Discussions, you solved a similar problem with $O((\log n) \log \log n)$ queries by first amplifying the success probability of answering each query and then running the usual binary search.

³This is related to the bonus question on the midterm, but you have to do a bit more here to make sure all pieces fit together in the last part of this problem.

⁴This is basically Problem 4, part (b) on the midterm, but make sure it works with your algorithm \mathcal{C} .