CS 6150: HW2

Due Date: Oct 1

This assignment has 7 questions, for a total of 100 points. Unless otherwise specified, complete and reasoned arguments will be expected for all answers.

Question	Points	Score
Tail Bounds I	20	
Tail Bounds II	15	
Medians	10	
Amplification	10	
The "median" trick	15	
Hashing with open addressing	15	
Derandomizing MAX CUT	15	
Total:	100	

Here are the three increasingly-strong inequalities we use to analyze the tail of a distribution.

Lemma 0.1 (Markov). *Let X be a random variable taking nonnegative values. Then for any $a > 0$,*

$$\Pr(X \geq a) \leq \frac{E[X]}{a}$$

Lemma 0.2 (Chebyshev). *Let X be a random variable with bounded variance. Then*

$$\Pr(|X - E[X]| \geq a) \leq \frac{Var[X]}{a^2}$$

Lemma 0.3 (Chernoff). *Let $X_1, \ldots, X_n$ be independent random variables taking the values $0, 1$ with $E[X_i] = \Pr(X_i = 1) = p_i$. Let $\mu = \sum_i p_i$. Then for any $\delta > 0$,*

$$\Pr(X \geq \mu(1+\delta)) \leq \exp(-\frac{\mu\delta^2}{3})$$

$$\Pr(X \leq \mu(1-\delta)) \leq \exp(-\frac{\mu\delta^2}{2})$$

Question 1: Tail Bounds I ..... **[20]**

(a) **[5]** The Markov inequality yields an *upper* bound on the probability of going far away from the mean. It is reasonable to ask whether we can do better. Show that you can't, by constructing some nonnegative random variable X and a value $a > 0$ such that

$$\Pr(X \geq a) = \frac{E[X]}{a}$$

(b) **[5]** Suppose we roll a fair die 100 times. Determine the probability that the sum of the rolls is at least 400 (using Markov's inequality) and the probability it is not in the range $[301, 399]$ (using Chebyshev's inequality).

(c) **[5]** Suppose we're given an algorithm that takes as input a string of n bits. We are told that the expected running time is $O(n^2)$ if the bits are chosen independently and uniformly at random. What can we say about the *worst-case* behavior of the algorithm on inputs of size n (using Markov's inequality).

(d) **[5]** We prove Chebyshev's inequality by applying Markov's inequality to the positive random variable $Y = (X - E[X])^2$. Can you generalize Chebyshev's inequality to higher moments of X (i.e values $E[X^k]$ for large k). In particular, set $k > 2$ to be some **even** number and derive a Chebyshev-like bound for the tail. What do you notice about such a bound ?

Question 2: Tail Bounds II ..... **[15]**

(a) **[5]** Repeat the Chebyshev analysis for the fair die above but using a Chernoff bound instead. Since your random variables are now not $0 - 1$, you will need a slightly different version of the Chernoff bound called a Hoeffding bound:
Lemma 0.4 (Hoeffding). *Let $X_1, \ldots, X_n$ be independent random variables where $a_i \leq X_i \leq b_i$. Let $S = \sum X_i$. Then*

$$\Pr(|S - E[S]| \geq t) \leq 2\exp(-\frac{2t^2}{\sum_i (b_i - a_i)^2})$$

(b) **[10]** Suppose you toss $2n$ fair coins and compute the difference between the number of heads and tails. On average, this difference is zero. But what is the probability that the difference is more than t ? (**Hint:** Consider just the number of heads).

Question 3: Medians ..... **[10]**

We saw in class that taking a median of three samples suffices to obtain a good approximation to the rank of the median. Assume that our goal is to return an element whose with rank r such that $|r - n/2| \leq \epsilon n/2$. Assume also that we desire this result with probability $1 - \delta$.

We can take a sample of size t and select the median as our result. What value of t must we pick (as a function of ϵ, δ and n) ? Do you notice something interesting/unusual ?

Question 4: Amplification ..... **[10]**

Consider a decision problem f (i.e one where the output is either zero or one). Suppose we are given a randomized algorithm A that on input x has the following properties:

- If $f(x) = 0$, then $A(x) = 0$
- If $f(x) = 1$, then $A(x) = 1$ with probability at least $2/3$.

Such an algorithm is said to have *one-sided error*, and as we've discussed in class we can amplify the probability of being correct merely by repeating it a number of times, and returning 1 if we ever see a 1, returning 0 otherwise.

But suppose our algorithm instead had the following properties:

- If $f(x) = 0$, then $A(x) = 0$ with probability at least $2/3$
- If $f(x) = 1$, then $A(x) = 1$ with probability at least $2/3$.

In other words, it has *two-sided* error rather than one-sided error.

(a) **[5]** What is the probability of success (in either case) if we implement the above procedure (returning a 1 if we ever see a 1, else returning a zero) after k iterations ?

(b) **[5]** What is the probability of success (in either case) of the following procedure: Run the algorithm k times and return the majority answer.

Question 5: The "median" trick ..... **[15]**

Suppose we have an algorithm that generates independent samples $X_1, \ldots$ of a random variable X. A natural estimate of the value of X is the mean $S = \sum_{i=1}^{t} X_i/t$ for some value of t. What is the probability that S is close to $E[X]$? Specifically, what is the probability that $|S - E[X]| \leq \epsilon E[X]$?

We can analyze this if we know that $\mathrm{Var}[X]$ is bounded. In particular, set $r = \sqrt{\mathrm{Var}[X]}/E[X]$.

(a) **[5]** Show that if we set $t = O(r^2/\epsilon^2\delta)$ then the probability is at least $1 - \delta$ (or in other words the probability of the difference *exceeding* $\epsilon E[X]$ is at most δ).

(b) **[2]** Suppose we only want a *weak estimate*: an estimate $\tilde{X}$ that is within $\epsilon E[X]$ with probability $2/3$. Show that in this case we need only set $t = O(r^2/\epsilon^2)$.

(c) **[8]** Now take the *median* of $O(\log(1/\delta))$ such weak estimates. Prove that this estimator does have the desired property of being within $\epsilon E[X]$ of the true estimate with probability $1 - \delta$. Note that we have done two things.

1. We've reduced the number of samples needed from $r^2/\epsilon^2\delta$ to $r^2 \log(1/\delta)/\epsilon^2$, which is an exponential reduction in dependence on $1/\delta$.
2. We've achieved a strong bound without using full Chernoff-like assumptions: we only needed X to have bounded variance.

Question 6: Hashing with open addressing ..... **[15]**

We've seen how to use random hash function to ensure only a small number of collisions in a hash bucket. We stored the overflow items in a *chain*, which is why we wanted to minimize its length. Such a hashing scheme is called *chaining*.

An alternate approach to hashing is called *open addressing*. Suppose we want to store n items in an array. We maintain an array of $2n$ slots, and also construct a hash function $h : U \times \mathbb{Z} \to [1 \ldots 2n]$. For each key k, the sequence $h(k, 0), h(k, 1), \ldots$ defines a *probe sequence* as follows.

- When k appear, the algorithm first tries to place it in $h(k, 0)$. If that entry is full, it tries $h(k, 1)$. If that is full, it tries $h(k, 2)$ and so on.
- When searching for an item q, we search the entries $h(q, i), i = 0, 1, \ldots$ until we either find q or find an empty spot (which means that q was never in the set).

Assume that $h(k, j)$ is uniform over $[1 \ldots 2n]$ for any k and that all $h(k, i)$ are independent.

(a) **[5]** Show that the probability an insertion takes more than r steps is at most 2^{-r}.

(b) **[5]** Show that for the i^{th} insertion ($i = 1, 2, \ldots, n$), the probability that more than $2 \log n$ probes are needed is $1/n^2$.

(c) **[5]** Let X_i be the number of probes needed to insert item i, and let $X = \max X_i$. Show that $\Pr(X > 2 \log n) \leq 1/n$.

Question 7: Derandomizing MAX CUT ..... **[15]**

While randomness is an easy way to design an algorithm, random bits are an expensive resource, and we'd like to use as few of them as possible. The process of removing randomness from an algorithm is called *derandomization*. There are three basic approaches to derandomization: using weak randomness, eliminating randomess by brute force, and the method of conditional expectations. We will examine the latter two methods here.

(a) **[5]** Suppose we have a randomized algorithm that runs in time $T(n)$, uses $O(\log n)$ random bits and returns the minimum of some function with a probability of success greater than $2/3$. Construct a deterministic algorithm that solves the same problem correctly in time $T'(n)$, expressed in terms of $T(n)$ and n (**Hint:** this is called the method of brute force).

The other approach works as follows. Suppose our randomized algorithm A produces some answer whose expectation is μ. We can imagine the algorithm as a sequence of deterministic operations punctuated by coin tosses $r_1, r_2, \ldots, r_m$. The value output by A is a function of the input and these coin tosses: ignoring the input, we can write the output value as $A(r_1, \ldots, r_m)$, and so $E[A(r_1, \ldots, r_m)] = \mu$

Consider the first coin toss r_1. We can write

$$E[A(r_1, \ldots, r_m)] = (1/2)E[A(0, r_2, \ldots, r_m)] + (1/2)E[A(1, r_2, \ldots, r_m)]$$

This is because the probability of $r_1 = 1$ is $1/2$. Notice however that one of the two expectations $E_0 = E[A(0, r_2, \ldots, r_m)]$ and $E_1 = E[A(1, r_2, \ldots, r_m)]$ must be at least as large as μ (since $\mu = (E_0 + E_1)/2$). Suppose we can compute these two numbers. Then we can *deterministically* pick the value of r_1 and we are sure that the final answer we get is no worse than the expected value, while having used one less random bit.

Repeating this over and over again yields a final result in which all bits are picked deterministically, and yet the final answer is at least as large as μ. Thus, we've derandomized the algorithm without paying a penalty. The catch is of course the estimation of E_0 and E_1.

Now consider the randomized algorithm for MAX CUT that yields a 0.5-approximation: Pick a vertex and label it as being black or white with equal probability. Repeat for all vertices, and return the partition into black and white as the desired cut.

(b) **[10]** Derandomize this algorithm using the method of conditional expectations. What is the resulting deterministic algorithm ? (**Hint:** think about how we might deterministically decide which color to give to a vertex based on the graph structure and how it influences E_0, E_1.)