

Problem Set 2

CS265/CME309, Winter 2026

Due: Friday 1/23, 11:59pm on Gradescope

Please follow the homework policies on the course website.

Note: Throughout this problem set (and in general in this class unless otherwise stated), you are allowed to use as a black box anything we have stated in the lecture notes.

1. **(10 pt.) [Approximating Frequencies.]** Suppose that A is a list of length n , containing elements from a large universe \mathcal{U} . Our goal is to estimate the frequencies of each element in \mathcal{U} : that is, for $x \in \mathcal{U}$, how often does x appear in A ? The catch is that A is too big to look at all at once. Instead, we see the elements of A one at a time: $A[0], A[1], A[2], \dots$. Unfortunately, \mathcal{U} is also really big, so we can't just keep a count of each element.

In this problem, we'll construct and analyze a randomized data structure that will keep a "sketch" of the list A , use small space, and will be able to efficiently answer queries of the form "approximately how often did x occur in A "?

Specifically, our goal is the following: we would like a (small-space) data structure, which supports operations `update`(x) and `count`(x). The `update` function inserts an item $x \in \mathcal{U}$ into the data structure. Given $\epsilon, \delta > 0$, the `count` function should have the following guarantee: After calling `update` n times, `count`(x) should satisfy

$$C_x \leq \text{count}(x) \leq C_x + \epsilon n \tag{1}$$

with probability at least $1 - \delta$, where C_x is the number of times that x occurs in A .

- (a) **(3 pt.)** Consider the following strategy. We start with an array R of length b initialized to all zeroes, and a random hash function $h : \mathcal{U} \rightarrow \{0, 1, \dots, b-1\}$ drawn from a *universal hash family* \mathcal{H} : Assume that for any $x \neq y$, $\Pr[h(x) = h(y)] = 1/b$.¹

Then the operations are:

- `update`(x): Increment $R[h(x)]$ by 1.
- `count`(x): Return $R[h(x)]$.

For each $i = 1, 2, \dots, n$, suppose that we call `update`($A[i]$). (This inserts all of A into the data structure). After we do this, what is the expected value of `count`(x)? Your answer should be in terms of C_x, n , and b , and you should justify your answer.

- (b) **(3 pt.)** Show that there is a choice of b that is $O(1/\epsilon)$ so that, for any fixed $x \in \mathcal{U}$, we have

$$\Pr[\text{count}(x) < C_x] = 0$$

and

$$\Pr[\text{count}(x) \geq C_x + \epsilon n] \leq \frac{1}{e}.$$

Note that this establishes (1) with probability at least $1 - 1/e$.

[**HINT:** *The first of the requirements is true no matter what b is.*]

[**HINT:** *For the second requirement, use Markov's inequality and take advantage of the first requirement.*]

¹Recall from CS161 or elsewhere that a universal hash family \mathcal{H} satisfies $\Pr_{h \in \mathcal{H}}[h(x) = h(y)] \leq 1/b$. For simplicity, in this problem we are saying that this is an equality, although the inequality is fine for the data structure to work.

(c) **(4 pt.)** In order to boost the success probability from $1 - 1/e$ to $1 - \delta$ for arbitrarily small δ , we will repeat the strategy from part (a) T times independently, so we keep T arrays R_1, \dots, R_T , and T independently chosen hash functions h_1, \dots, h_T .

- i. Fill in the details: How do you implement **update** and **count** for this T -repeated version of the data structure?
- ii. How big do you need to take T so that (1) is satisfied with probability at least $1 - \delta$?
- iii. With this value of T , how much space does the data structure use? Asymptotic notation is fine. You may assume that it takes $O(\log |\mathcal{U}|)$ bits to store a hash function h from \mathcal{H} . (And recall that it takes $O(\log |\mathcal{U}|)$ bits to store an element of \mathcal{U} , and $O(\log n)$ bits to store an integer in $\{0, 1, \dots, n\}$).

2. **(12 pt.) [Median-of-means]** Let X_1, X_2, \dots, X_n be i.i.d. samples of a random variable X with $\mathbb{E}[X] = \mu$ and $\text{Var}[X] \leq 1$.

Suppose that k divides n , let $m = n/k$. Consider the following algorithm for estimating μ from n samples. First, we divide the n samples into k groups of size m :

$$\begin{aligned} A_1 &= \{X_1, \dots, X_m\} \\ A_2 &= \{X_{m+1}, \dots, X_{2m}\} \\ &\vdots \\ A_k &= \{X_{(k-1)m+1}, \dots, X_n\}. \end{aligned}$$

Then let Y_i be the mean of A_i for each $i = 1, \dots, k$. Finally, let

$$\hat{\mu} = \text{Median}(Y_1, \dots, Y_k).$$

Let $\epsilon, \delta \in (0, 1)$. Our goal in this problem will be to pick k and m (and hence $n = mk$) in terms of ϵ and δ so that

$$\Pr[|\hat{\mu} - \mu| > \epsilon] \leq \delta.$$

- (a) **(3 pt.)** Show that $\Pr[|Y_j - \mu| > \epsilon] \leq \frac{1}{m\epsilon^2}$.
- (b) **(6 pt.)** Say that $j \in \{1, 2, \dots, k\}$ is “good” if $|Y_j - \mu| \leq \epsilon$, and “bad” otherwise. Let B be the number of bad j ’s. Suppose that $m = 4/\epsilon^2$. Use Chebyshev’s inequality to show that

$$\Pr[B \geq k/2] \leq \frac{4}{k}.$$

[HINT: First show that, with this choice of m , $\Pr[B \geq k/2] \leq \frac{16\text{Var}(B)}{k^2}$]

- (c) **(3 pt.)** Using the previous parts, how should we pick k and m in terms of ϵ and δ so that $\Pr[|\hat{\mu} - \mu| > \epsilon] \leq \delta$? Explain why your choice is correct. What is the final sample complexity (that is, n), in terms of ϵ and δ ?
- (d) **(0 pt.) [This part is optional and won’t be graded.]** What might be the advantages of taking a median of means as in this problem, rather than letting $\hat{\mu} = \frac{1}{n} \sum_{i=1}^n X_i$ be the mean of all the samples?

3. (8 pt.) [Tightness of Markov's and Chebyshev's Inequalities]

- (a) (4 pt.) Show that Markov's inequality is tight. Specifically, for each value $c > 1$, describe a distribution D_c supported on non-negative real numbers such that if the random variable X is drawn according to D_c then
- (1) $\mathbb{E}[X] > 0$, and
 - (2) $\Pr[X \geq c\mathbb{E}[X]] = 1/c$.
- (b) (4 pt.) Show that Chebyshev's inequality is tight. Specifically, for each value $c > 1$, describe a distribution D_c supported on real numbers such that if the random variable X is drawn according to D_c then
- (1) $\mathbb{E}[X] = 0$ and $\text{Var}[X] = 1$, and
 - (2) $\Pr[|X - \mathbb{E}[X]| \geq c\sqrt{\text{Var}[X]}] = 1/c^2$.
- (c) (0 pt.) [One-sided version of Chebyshev's Inequality] [Optional: this question won't be graded.] Consider the following statement: Let X be a random variable with $\text{Var}[X] = 1$. For all $t > 0$,

$$\Pr[X - \mathbb{E}[X] \geq t] \leq \frac{1}{1 + t^2}.$$

- i. Prove this statement.
- ii. Show that this is tight: For any $t > 0$, come up with a random variable X with distribution D_t and variance 1 for which $\Pr[X - \mathbb{E}[X] \geq t] = \frac{1}{1+t^2}$.

4. (0 pt.) [This whole problem is optional and will not be graded.] In this problem, you'll analyze a different primality test than we saw in class. This one is called the *Agrawal-Biswas Primality test*.

Given a degree d polynomial $p(x)$ with integer coefficients, for any polynomial $q(x)$ with integer coefficients, we say $q(x) \equiv t(x) \pmod{(p(x), n)}$ if there exists some polynomial $s(x)$ such that $q(x) = s(x) \cdot p(x) + t(x) \pmod n$. (Here, we say that $\sum_i c_i x^i = \sum_i c'_i x^i \pmod n$ if and only if $c_i = c'_i \pmod n$ for all i .) For example, $x^5 + 6x^4 + 3x + 1 \equiv 3x + 1 \pmod{(x^2 + x, 5)}$, since $(x^3)(x^2 + x) + (3x + 1) = x^5 + x^4 + 3x + 1 \equiv x^5 + 6x^4 + 3x + 1 \pmod 5$.

Agrawal-Biswas Primality Test.

Given n :

- If n is divisible by 2,3,5,7,11, or 13, or is a perfect power (i.e. $n = c^r$ for integers c and r) then output **composite**.
- Set d to be the smallest integer greater than $\log n$, and choose a random degree d polynomial with leading coefficient 1:

$$r(x) = x^d + c_{d-1}x^{d-1} + \dots + c_1x + c_0,$$

by choosing each coefficient c_i uniformly at random from $\{0, 1, \dots, n-1\}$.

- If $(x+1)^n \equiv x^n + 1 \pmod{(r(x), n)}$ then output **prime**, else output **composite**.

Consider the following theorem (you can assume this if you like, or for even more optional work, try to prove it!):

Theorem 1 (Polynomial version of Fermat's little theorem).

- If n is prime, then for any integer a , $(x-a)^n = x^n - a \pmod n$.
- If n is not prime and is not a power of a prime, then for any a s.t. $\gcd(a, n) = 1$ and any prime factor p of n , $(x-a)^n \not\equiv x^n - a \pmod p$.

First, show that if n is prime, then the Agrawal-Biswas primality test will always return **prime**.

Now, we will prove that if n is composite, the probability over random choices of $r(x)$ that the algorithm successfully finds a witness to the compositeness of n (and hence returns **composite**) is at least $\frac{1}{4d}$.

- (a) Using the polynomial version of Fermat's Little Theorem, and the fact that, for prime q , every polynomial over \mathbb{Z}_q that has leading coefficient 1 (i.e. that is "monic") has a unique factorization into irreducible monic polynomials, prove that the number of irreducible degree d factors that the polynomial $(x+1)^n - (x^n + 1)$ has over \mathbb{Z}_p is at most n/d , where p is any prime factor of n . (A polynomial is irreducible if it cannot be factored, for example $x^2 + 1 = (x+1)(x+1) \pmod 2$ is not irreducible over \mathbb{Z}_2 , but $x^2 + 1$ is irreducible over \mathbb{Z}_3 .)

[**HINT:** Even though this question sounds complicated, the proof is just one line...]

- (b) Let $f(d, p)$ denote the number of irreducible monic degree d polynomials over \mathbb{Z}_p . Prove that if n is composite, and not a power of a prime, the probability that $r(x)$ is a witness to the compositeness of n is at least $\frac{f(d,p)-n/d}{p^d}$, where p is a prime factor of n .

[**HINT:** p^d is the total number of monic degree d polynomials over \mathbb{Z}_p .]

- (c) Now complete the proof, and prove that the algorithm succeeds with probability at least $1/(4d)$, leveraging the fact that the number of irreducible monic polynomials of degree d over \mathbb{Z}_p is at least $p^d/d - p^{d/2}$. (You should be able to prove a much better bound, though $1/4d$ is fine.)

[**HINT:** You will also need to leverage the fact that we chose $d > \log n$ and also explicitly made sure that n has no prime factors less than 17.]