

CS 49/249: Randomized Algorithms (Spring 2021) Problem Set

This is an alive document which will refresh frequently. Check every weekend for newer problems.

Can be done in groups of size ≤ 2 .

Instructions

- *Credit Statements*: At the beginning of each problem you must write who all (including the teaching staff) you discussed this problem set with. This is **important**. Even if you did not talk with anyone about any of the problems, you need to mention “No one”. *Without a credit statement, you may be awarded 0 for the p-set.*
 - *External Sources*: You are not allowed to consult any sources other than the notes and the text-book. Many of these problems have solutions out there on the web. Don’t go hunting for them. If you stumble upon them, then cite it. Uncited solutions will be an **honor code violation**.
 - *Presentation*: Your presentation should satisfy the following three C’s: they should be *Correct, Clear, Concise*. Do not *ramble*. Ideally, I should be able to read and completely understand any answer to a single problem in **less than five minutes**.
 - *LaTeX*: You **have** to use LaTeX. Style files will be provided in Canvas. The main reason is that I will often choose the best solution among you all as official solutions.
 - This Problem Set is worth 30% of your total points. I will usually post 4-5 problems per week. To get these, you must complete at least 60% of the (“normal” non-🧠) problems. Please hand in work often to obtain feedback from the teaching staff.
 - Some of the problems are possibly challenging (some of them I don’t know the answer to myself) and are marked with 🧠. Doing these will count towards your “engagement” in the course. Instructive problems which teach you a concept/algorithm are marked with ❤️.
 - **News (May 8th)**: Some problems have been greyed out as their solutions have been released. Do not attempt them anymore.
-

Week 1

Problem 1 (Checking Matrix Multiplication). ♥

Let A, B, C be three $n \times n$ matrices. You need to decide if $C = AB$ where AB is the standard matrix multiplication. That is, AB is an $n \times n$ matrix M whose (i, j) th entry is $M_{ij} = \sum_{k=1}^n A_{ik}B_{kj}$. In other words, M_{ij} is the dot product of the i th row of A and j th column of B . The naive way to multiply two $n \times n$ matrix takes $O(n^3)$ time, and to date no algorithm faster than $O(n^{2.37})$ is known.

Design and analyze a Monte-Carlo randomized algorithm to check if $AB = C$ which runs in $O(n^2)$ time.

Problem 2 (The Schwartz-Zippel Lemma). ☹ ♥

Let $p(x_1, x_2, \dots, x_n)$ be a **non-zero** polynomial on n variables with total degree d . So, for instance the polynomial $x_1^2x_2^2 + x_3^2 + x_1x_2x_3$ is a polynomial on 3 variables with total degree 4. Let S be an arbitrary collection of numbers. Let r_1, r_2, \dots, r_n be n numbers chosen from S independently and uniformly at random. Then, prove

$$\Pr[p(r_1, r_2, \dots, r_n) = 0] \leq \frac{d}{|S|}$$

Here is a skeleton of the proof.

- First establish the case for $n = 1$.
- Next, write $p(x_1, \dots, x_n)$ as $x_1^k Q_k(x_2, \dots, x_n) + x_1^{k-1} Q_{k-1}(x_2, \dots, x_n) + \dots + Q_0(x_2, \dots, x_n)$. That is, we pick out the powers of x_1 separately. k is the largest power of x_1 appearing in p .
 - What is the total degree of Q_k ? What does induction tell you about $\Pr[Q_k(r_2, \dots, r_n) = 0]$?
 - Finish up the proof using induction.

Remark: This lemma gives a fast **randomized** algorithm to check if a polynomial, which we can only access via an evaluation oracle, is equivalent to the 0 polynomial or not : simply evaluate the polynomial on random points picked from a modestly sized set. We do not know of any efficient deterministic algorithm for this : it is a fundamental open question in the theory of computation!

Problem 3 (Quick Select).

In this problem we look at an algorithm to find the k th smallest number of an unsorted array $A[1 : n]$. k could be as large as $n/2$. We are interested in the *number of comparisons* made by the algorithm. Before reading further, answer this: what's the algorithm for this problem you know right now which makes the smallest number of comparisons?

```

1: procedure QUICKSELECT( $A[1 : n], k$ ):
2:    $\triangleright$  Returns the  $k$ th smallest number in  $A[1 : n]$ .
3:   if  $k = 1$  then:
4:     Find the minimum of  $A[1 : n]$  making  $\leq n$  comparisons.
5:   while true do:
6:      $q \leftarrow \text{RANDOM}(A[1 : n]) \triangleright$  A random element of  $A[1 : n]$ 
7:      $(A_1, A_2) = \text{PIVOT}(A[1 : n], q) \triangleright$   $A_1$  is an array of all elements in  $A[1 : n]$  strictly smaller than  $q$ ;  $A_2$  are the rest.
8:     if  $n/3 \leq \text{len}(A_1) \leq 2n/3$  then:
9:       break the While Loop
10:     $\triangleright$  Convince yourself: at this point both  $A_1$  and  $A_2$  have length in  $[n/3, 2n/3]$ 
11:    if  $\text{len}(A_1) \geq k$  then:
12:      QUICKSELECT( $A_1, k$ )
13:    else:
14:      QUICKSELECT( $A_2, k - \text{len}(A_1)$ )

```

- On taking input an array $A[1 : n]$ and $k > 1$, what is the *expected* number of comparisons made in the While Loop (Lines: 5-9). You can give as tight an upper bound as you can : we are interested in the coefficient of n .
- Suppose $T(n)$ is the expected worst case number of comparisons QUICKSELECT makes on any array of length $\leq n$. That is,

$$T(n) := \max_{A[1:n]} \mathbf{Exp}[T_Q(A)]$$

where $T_Q(A)$ is the number of comparisons on array A . Write the recurrence inequality which gives an *upper bound* on $T(n)$ in terms of $T(m)$ for some m which is significantly smaller than n . Solve the recurrence and write $T(n)$. We are interested in the *coefficient* of n , so Big-Oh notation won't do.

Problem 4 (A question of expectation). 

Remember longest increasing subsequences? Given an array $A[1 : n]$, an increasing subsequence of length k is a collection $1 \leq i_1 < i_2 < \dots < i_k \leq n$ such that $A[i_1] \leq A[i_2] \leq \dots \leq A[i_k]$. The LIS of $A[1 : n]$ is the length of the longest increasing subsequence.

Now, let $A[1 : n]$ be a *random* permutation of the numbers $\{1, 2, \dots, n\}$. Prove that $\mathbf{Exp}[\text{LIS}(A[1 : n])] = \Theta(\sqrt{n})$. Note there are two things to show, and both are interesting.

Problem 5 (Number of Approximate Minimum Cuts). 

In an undirected multigraph $G = (V, E)$, a cut $S \subseteq V$ is an α -approximate minimum cut, for some integer $\alpha \geq 1$, if $|\partial S| \leq \alpha \cdot C$ where C is the value of the minimum cut in G . Prove that the number of α -approximate minimum cuts is $O(n^{2\alpha})$.

Week 2

Problem 6 (A curious “boosting” situation).

Let N be a statistic for which someone constructs an estimate $\widehat{\text{est}}$ with the following curious property:

$$\Pr[\widehat{\text{est}} \leq (1 - \varepsilon)N] \leq 0.01 \quad \text{but} \quad \Pr[\widehat{\text{est}} \geq (1 + \varepsilon)N] \leq 0.9$$

That is, the probability that $\widehat{\text{est}}$ is an “ ε -under-estimate” is small, less than 1%, but the probability that $\widehat{\text{est}}$ is an “ ε -over-estimate” is only less than 90%. Note, we are not insisting $\widehat{\text{est}}$ is an unbiased estimator, nor are we saying anything about its variance.

Show how to obtain an (ε, δ) -multiplicative estimator est using independent samples of $\widehat{\text{est}}$. How many samples do you need (your answer should be a function of δ . Answer in Big-Oh fine)?

Problem 7. ♥

You have a fair coin which tosses heads or tails with probability $1/2$ -each. n is a parameter. You keep tossing this coin stopping till you see n heads. Let Z be the total number of tosses you have done so far. This is a random variable.

- What is $\text{Exp}[Z]$?
- For any $\varepsilon \in (0, 1)$, what is the best upper bound on the probability $\Pr[Z \notin (1 \pm \varepsilon) \text{Exp}[Z]]$? Express your answer as a function of n and ε .

Problem 8 (A question of fairness).

Let U be a universe of n elements, where n is even. Let $A \subseteq U$ be colored *azure* and the remaining elements $B := U \setminus A$ be colored *blue*. Assume $|A| = |B| = \frac{n}{2}$. Sample a random subset R of U as follows: independently, for every element $i \in U$, you add it to R with probability $\frac{s}{n}$. Define the “unfairness” of your sample R defined as $\Delta_R := \left| |R \cap A| - |R \cap B| \right|$ to be the difference in the number of azure and blue elements in your sample. For what value of t (as a function of s , δ , and n), can you prove that the probability $\Pr[\Delta_R \geq t]$ becomes smaller than δ ? Big-Oh notation is fine.

Problem 9 (Randomized Selection).

To see if you have understood the RANDOMIZED-MEDIAN algorithm, show how you will modify the algorithm and analysis to find the k th smallest item in A . That is, the item x with $\text{rank}_A(x) = k$. Your algorithm should abort with probability $\leq n^{-O(1)}$, and should make $\leq n + k + o(n)$ comparisons.

Problem 10 (QuickSort Analysis). ☹

Prove that QUICKSORT, as we saw in Lecture 2, actually satisfies the following: it makes $O(kn \log n)$ comparisons with probability $\geq 1 - \frac{1}{n^k}$. Here is a skeleton of an argument which may help.

Fix an element $a \in A$ and let T_a be the number of comparisons made by QUICKSORT where one of the entries is a . The total number of comparisons is $\sum_{a \in A} T_a$.

- a. Let P_1, P_2, \dots, P_T be the arrays such that (a) $a \in P_t$, and (b) $\text{QUICKSORT}(P_t)$ is called. So, $P_1 = A$ the original array. P_2 is either A_1 or A_3 (depending on the pivot q), or there is no P_2 if $q = a$ in which case $a \in A_2$. Note that P_t 's and T are random variables.

Express T_a as a function of these P_t 's and T .

- b. For $1 \leq t \leq T$, define a random variable Y_t as follows. Note that QUICKSORT breaks the array A into three sub-arrays A_1, A_2, A_3 . We set $Y_t = 1$ if $q = a$ or if $\frac{|A|}{3} \leq |A_1| \leq \frac{2|A|}{3}$. Note that $Y_t = 1$ implies $t = T$ or $|P_{t+1}| \leq \frac{2|P_t|}{3}$.

What is $\Pr[Y_t = 1]$? Are these independent random variables?

- c. Consider the chain Y_1, Y_2, \dots, Y_T : again remember T is a random variable. How many of these Y_t 's can be 1? Note that every time $Y_t = 1$, we have $|P_{t+1}| \leq (2/3)|P_t|$.
- d. What is the probability $\Pr[T > 10 \lg_2 n]$?
- e. Prove that QUICKSORT makes $\leq 10n \log_2 n$ comparisons with probability $1 - n^{-O(1)}$.

Week 3

Problem 11 (Implementing Karger's RANDMC Algorithm). ♥

Let $G = (V, E)$ be an undirected multigraph on n vertices. Note that $|E|$ could be much larger than $\binom{n}{2}$. Suppose you have the two data-structures: (a) for every vertex, you have a dictionary $\text{deg}(v)$ which counts the number of edges incident on v , and (b) you have an $n \times n$ array which maintains $A_{uv} = A_{vu}$, the number of edges between u and v . Using this show how to sample a random edge in E in $O(n)$ time.

Now suppose you picked $e = (x, y)$ and you decide to contract it. That is, G will get rid of x and y , and add a new vertex (xy) . What are the changes you need to make to the data-structures? Show that all this can be done in $O(n)$ time (assuming addition/subtraction/multiplication/division are all $O(1)$ time operations). Using this argue that RANDMC takes $O(n^2)$ time.

Problem 12 (Taking care of false positives). ♥

Let U be a universe of n elements where every element $e \in U$ has a score $\text{score}(e)$. Let $S := \sum_{e \in U} \text{score}(e)$. Imagine you have an importance sampling algorithm \mathcal{A} which return an $e \in U$ with probability proportional to $\text{score}(e)$. That is, $\Pr[\mathcal{A} \text{ returns } e] = \frac{\text{score}(e)}{S}$

Design an algorithm which takes a parameter $K, \varepsilon \in (0, 1), \delta \in (0, 1/2)$, makes $N := O\left(\frac{S}{K} \cdot \frac{\ln(n/\delta)}{\varepsilon^2}\right)$ calls to \mathcal{A} and returns a subset $H \subseteq U$ such that with probability $\geq (1 - \delta)$ we have both:

- (a) every element $e \in U$ with $\text{score}(e) \geq K$ is present in H , that is, total recall.
- (b) every element $e \in H$ has $\text{score}(e) \geq K(1 - \varepsilon)$, that is, high precision.

Remark: When U is the collection of all pairs in $[n] \times [n]$, $\text{score}(i, j) = (A^T A)_{ij}$, and the algorithm \mathcal{A} is the Cohen-Lewis sampler, then this shows how to get rid of false positives.


Problem 13 (Estimating the number of connected components using graph queries). *Broken into parts to make it simpler. Please first try without looking at hints.*

We assume the undirected graph access model used in the lecture notes. To recall, you can access a random vertex v , and for any vertex v you can query its degree $\text{deg}(v)$, and for any integer $1 \leq i \leq \text{deg}(v)$, you can obtain the i th neighbor of v (in some arbitrary but fixed order). In this exercise the objective is to estimate the number of connected components in the graph upto an *additive* error of εn . That is, if C^* is the number of connected components, we wish to return an estimate C such that $\Pr[|C - C^*| \geq \varepsilon n] \leq \delta$. Proceed in the following way. First, observe that we want to obtain an (ε, δ) -additive estimate to $N := \frac{C^*}{n}$. Next,

- a. Define $\text{conn}(v)$ to be the size of the connected component containing v . Establish a relation between C^* and $\sum_{v \in V} \frac{1}{\text{conn}(v)}$.
- b. Consider the following $\widehat{\text{est}}$: sample a random vertex v , and set $\widehat{\text{est}} := \frac{1}{\text{conn}(v)}$. What is $\mathbf{Exp}[\widehat{\text{est}}]$? What is $\mathbf{Var}[\widehat{\text{est}}]$? How many samples do we need to get an (ε, δ) -additive estimate to $\frac{C^*}{n}$?
- c. A **big** issue: how do we figure out $\text{conn}(v)$? If we run DFS from v and the whole graph is connected, we may take $O(n + m)$ queries. So, instead define $\text{conn}'(v) := \min\left(\text{conn}(v), \frac{2}{\varepsilon}\right)$. Show how $\text{conn}'(v)$ can be evaluated in $O(1/\varepsilon^2)$ queries.

d. Prove that $\left| \sum_{v \in V} \frac{1}{\text{conn}'(v)} - \sum_{v \in V} \frac{1}{\text{conn}(v)} \right| \leq \frac{\varepsilon n}{2}$.

e. Complete all the details to give an (ε, δ) -additive approximation to $\frac{C^*}{n}$ making $O\left(\frac{1}{\varepsilon^4} \ln(1/\delta)\right)$ queries.

Problem 14 (Feige's Estimate of Average Degree). 

In this exercise, we will sketch the steps needed to show how to obtain, with probability $\geq 1 - \delta$, a $(2 + \varepsilon)$ -approximate estimate of d_{avg} by sampling $N = O\left(\sqrt{\frac{n}{d_0}} \cdot \text{poly}\left(\frac{1}{\varepsilon}\right) \ln(1/\delta)\right)$ random vertices in a graph and querying their degrees. Here d_0 is a known lower bound on d_{avg} . Note that this procedure won't use neighbor queries.

First we order the vertices in decreasing order of degree: $d_1 \geq d_2 \geq \dots \geq d_n$, where d_i is the degree of the i th vertex in this order.

a. Prove that for any $1 \leq k \leq n$, we have

$$\sum_{i < k} d_i \leq \sum_{i \geq k} d_i + 2 \binom{k}{2} \quad (1)$$

This is the *crucial* fact that will use that these d_i 's are not arbitrary numbers but arise as degrees of graphs.

b. Recall that $\widehat{\text{est}}$ is the degree of a random vertex. Let $k := \lceil \sqrt{\varepsilon n d_{\text{avg}}} \rceil$ and define a random variable X which equals $\widehat{\text{est}}$ if $\deg(v) \leq d_k$, but is 0 otherwise. Note, this random variable is for analysis purpose only since neither do we know d_{avg} nor do we know d_k . What is important is that $X \leq \widehat{\text{est}}$ with probability 1.

(i) What is $\mathbf{Exp}[X]$? Using (1), prove $d_{\text{avg}} \geq \mathbf{Exp}[X] \geq \left(\frac{1-\varepsilon}{2}\right) \cdot d_{\text{avg}}$.

(ii) Prove that $d_k \leq \sqrt{\frac{n d_{\text{avg}}}{\varepsilon}}$.

(iii) Prove that $\frac{\mathbf{Var}[X]}{\mathbf{Exp}^2[X]} = O\left(\sqrt{\frac{n}{d_0}}\right)$. Now conclude that if we take $N_1 := O\left(\sqrt{\frac{n}{d_0}} \cdot \frac{1}{\varepsilon^3}\right)$ samples of $\widehat{\text{est}}$ and take the average to get est , then we get that

$$\mathbf{Pr}\left[\text{est} < \left(\frac{1-\varepsilon}{2}\right) \cdot d_{\text{avg}}\right] \leq \frac{\varepsilon}{12} \quad (2)$$

That is, est can't be much smaller than $d_{\text{avg}}/2$, whp.

c. We still haven't ruled out the fact that est could be much *bigger* than d_{avg} . To take care of this, we simply use Markov's inequality. Note that $\mathbf{Pr}[\widehat{\text{est}} > (1 + \varepsilon)d_{\text{avg}}] \leq \frac{1}{1+\varepsilon} < \left(1 - \frac{\varepsilon}{2}\right)$ if $\varepsilon < 1$. Use this to argue that the est obtained above by taking average of N_1 samples satisfies

$$\mathbf{Pr}[\text{est} > (1 + \varepsilon)d_{\text{avg}}] \leq 1 - \frac{\varepsilon}{2} \quad (3)$$

d. Finish up using ideas in Problem 6. Note (2) and (3) are similar to that problem's premise. How many vertices do you need to sample in all as a function of n, d_0, ε , and δ ?

Week 4

Problem 15 (General Maximum Load : an exercise in using the general Chernoff bound).

Imagine the balls-and-bins process with m balls and n bins. Find as general a function $f(m, n)$ as you can such that you assert that with probability $\geq 1 - \frac{1}{n}$, the maximum load in any bin is $\leq f(m, n)$. When $m = n$, the lecture notes assert $f(m, n) = O\left(\frac{\ln n}{\ln \ln n}\right)$. We are interested in the form of $f(m, n)$ and not the constants. So, an answer in the Big-Oh notation is not only OK, it is encouraged.

After you have obtained this $f(m, n)$, write it as a univariate function of n in the special cases of $m = \sqrt{n}$, $m = n^{2/3}$, $m = n \ln n$, and $m = n^2$.

Problem 16 (Playing with Poisson Random Variables). Let $Z \sim \text{Pois}(k)$ where k is a positive integer.

- Prove that $\text{Var}[Z] = k$. In fact this holds even when k is not an integer.
- Prove that a **mode** of Z is k as well. That is, $\Pr[Z = k] \geq \Pr[Z = j]$ for any other integer j .
- 🌀 Find a constant c such that $\Pr[Z \geq k] \geq c$ and $\Pr[Z \leq k] \geq c$. What is the *largest* c you can find? The true answer is $c = \frac{1}{2}$, and extra admiration if you can achieve this.

Problem 17 (Number of empty bins.). ♥

Suppose you throw n balls independently into n bins uniformly at random. Let X denote the number of **empty** bins at the end of the experiment. We wish to understand how this looks like.

- What is $\text{Exp}[X]$? Your answer should be a function of n . When $n \rightarrow \infty$, what does your answer converge to?
- What is the best upper bound you can give to the probability $\Pr[X \geq \frac{n}{2}]$ and $\Pr[X \leq \frac{n}{3}]$?

Problem 18 (Coupon Collector's Lower Bound). ♥

Recall the coupon collector problem: there are n coupons, and every day you receive one of them uniformly at random. Let X be the number of days when you receive all n coupons. In the lecture, we proved $\text{Exp}[X] = nH_n$. For any positive $c > 0$, prove

$$\Pr[X \leq n \ln n - cn] \leq 2e^{-e^c}$$

Remark: Actually, it is not much more difficult to show that if c is a fixed constant, then the probability of the LHS tends to e^{-e^c} as $n \rightarrow \infty$.

Week 5

Problem 19 (Comparing Multisets).

In this problem you are given two *multisets* S_1 and S_2 from a universe $U = \{0, 1, \dots, N - 1\}$. That is, the same element may appear more than once in a set. You have to design an algorithm which decides if these two multisets are equal. That is, the number of times every element appears in the multiset is the same. Using universal hash functions, describe a randomized algorithm to solve this problem in expected $O(n)$ running time, where n is the number of elements in $|S_1| + |S_2|$ counting multiplicities.

Problem 20 (Is the b necessary?). ♥

In the lecture notes, the Carter-Wegman universal family takes two parameters a, b and sets $h_{a,b}(x) = ((ax + b) \bmod p) \bmod n$. Is the extra b necessary? That is, consider the family

$$F := \{h_a : a \in \{1, \dots, p - 1\}\} \text{ where } h_a(x) := ((ax) \bmod p) \bmod n$$

What is wrong with the following “proof” which shows that G is a UHF? In particular, show an example of $x \neq y$ where $\Pr_{h \in F}[h(x) = h(y)]$ is in fact $> \frac{1}{n}$. Using that, find the precise line which has a bug.

Proof. Fix $x \neq y$ in U . Note that for $h_a(x) = h_a(y)$, we must have that $ax \bmod p = ay \bmod p + k \cdot n$ for some integer k . Which in turn implies,

$$a \cdot (x - y) \bmod p = k \cdot n$$

Since the LHS is in $\{0, 1, \dots, p - 1\}$, we get $k \leq (p - 1)/n$.

Now, $a \cdot (x - y) \equiv_p k \cdot n$, we must have $a = (x - y)^{-1} \cdot kn \bmod p$. No other value would work.

Since k has $\leq \frac{p-1}{n}$ distinct values possible, at most $(p - 1)/n$ values of a could lead to $h_a(x) = h_a(y)$.

That is, the probability this occurs is $\leq 1/n$, since a is picked randomly from $\{1, 2, \dots, p - 1\}$. □

Correct the proof to give the best possible upper bound on $\Pr_{h \in F}[h(x) = h(y)]$ for $x \neq y$.

Problem 21 (Bloom Filters). ♥

Let U be a finite set of N objects (think all images in the world, if you will),. Let $P \subseteq U$ be a subset of images of interest (perhaps this class photo roster), and the size of $|P| = m \ll N$. We desire a data structure which allows two operations: **Insert**(x) which inserts the element $x \in U$ into P , and **Search**(u) which given an element $u \in U$ says YES if $u \in P$ and NO, otherwise. We wish to do so maintaining *very little* space. We are allowing *false positives*, that is, we are OK if you say $u \in P$ even when it is not, but we want to bound the probability of a false positive.

To do so, we maintain a *bit-array* $B[1 : n]$ initialized to all 0s for some natural number n which you have to figure out. Next, we *independently* select k hash functions h_1, h_2, \dots, h_k from the set of **all** functions. Crucially, not a UHF (see the ☹ question below). Using these, we implement the data structure operations as follows.

- **Insert**(x): For $1 \leq i \leq k$, set $B[h_i(x)] = 1$.
- **Search**(u): Say YES if *all* $B[h_i(u)]$ return 1; say NO even if a single $B[h_i(u)]$ returns 0.

Note that **Search**(u) will give no *false negatives*.

Consider an operation where the m elements of P have been **Insert**-ed. For any $j \in [n]$, let \mathcal{E}_j be the event that $B[j] = 1$ after the m insertions. That is, the j th bit of the bit-array is set to 1.

- a. Fix a j in $\{1, \dots, n\}$. What is $\Pr[\mathcal{E}_j]$? Use this to figure out p defined as the **expected** fraction of entries j with $B[j] = 1$ (or “non-empty bins”).
- b. Assuming that the p you calculated above is the **actual** fraction, calculate the *false positive rate* of **Search** in terms of n, m and k . That is, given an element $u \notin P$, what is the probability that **Search**(u) says YES?
- c. Suppose we fix n and m both. Figure out the k which minimized the false positive error. (This won't be an integer, but swing with it). For this value of k , write the probability of false positive as a function of n and m .
- d. How large does $\frac{n}{m}$ need to be to make this false positive error $< 1\%$?
- e. Finally, what does Poisson approximation tell you about the assumption you made? The assumption that the expected was actual. How does it change your calculation of part (d)?
- f. ☹ How do things change when h_i 's are drawn from the Carter-Wegman family mapping U to $[n]$?

Week 6

Problem 22 (Heavy Hitters). ♥

Suppose we wish to use the COUNT-MIN sketch to solve the following HEAVY HITTERS problem : given a parameter ϕ , we wish to (a) return all elements $i \in [n]$ such that $\mathbf{f}_i \geq \phi F_1$, but (b) we don't want to return any element $j \in [n]$ with $\mathbf{f}_j \leq (\phi - \varepsilon)F_1$. More precisely, we want both things (a) and (b) to occur with probability $1 - \delta$. Here $F_1 = \|\mathbf{f}\|_1$.

Note that there is a simple solution : run the COUNT-MIN algorithm with δ/n making sure that with probability $1 - \delta$, for **all** $i \in [n]$ we have $\mathbf{f}_i \leq \hat{\mathbf{f}}_i \leq \mathbf{f}_i + \varepsilon \cdot F_1$. And thus, we simply go over all $i \in [n]$ and return the set $S := \{i : \hat{\mathbf{f}}_i > (\phi - \varepsilon)F_1\}$. Convince yourself that (a) and (b) occur with probability $1 - \delta$.

However, the **time** taken to return the set is $O(n)$. Design an algorithm using the idea used in range counting to obtain an algorithm that takes much less time. It should be closer to $\log n$. You may assume $\frac{1}{\phi}$ is a constant.

Problem 23 (A better analysis of COUNT-SKETCH). ♥

In class, we showed the following estimate about COUNT-SKETCH: we proved that for any $i \in [n]$, the probability $\Pr \left[\left| \hat{\mathbf{f}}_i - \mathbf{f}_i \right| \geq \varepsilon \|\mathbf{f}\|_2 \right] \leq \frac{1}{3}$.

We can analyze better. Let H denote the set of elements i with the $\left\lceil \frac{1}{\varepsilon^2} \right\rceil$ -largest \mathbf{f}_i 's. Let $L := [n] \setminus H$ be the long tail of low frequency elements. Define $\|\mathbf{f}\|_{\text{tail}} := \sqrt{\sum_{i \in L} \mathbf{f}_i^2}$. Clearly, $\|\mathbf{f}\|_{\text{tail}} < \|\mathbf{f}\|_2$, and indeed in many applications can be much smaller. Prove that COUNT-SKETCH run with $k \leq \frac{10}{\varepsilon^2}$ actually satisfies

$$\Pr \left[\left| \hat{\mathbf{f}}_i - \mathbf{f}_i \right| \geq \varepsilon \|\mathbf{f}\|_{\text{tail}} \right] \leq \frac{1}{3}$$

for every $i \in [n]$.

Problem 24 (Estimating F_k with general updates in insertion only streams).

In the lecture, we estimate F_k assuming every update is of the form $(i, 1)$. Show how you will modify the algorithm if the updates looked like (i, c) where c is a positive integer. If this needs you to modify the reservoir sampling procedure, then show how you will do this. Does the number of samples change depending on how large these c 's are?

Week 7

Problem 25 (Approximate counting with smaller space). ♥

Imagine a stream of elements coming at you, and you want to count the number of items that are passing by. Clearly, this just requires maintaining a counter, and if m items pass by, then one needs only $\lceil \log_2 m \rceil$ bits. Can one do away with smaller space? The answer is yes, if one is OK with approximation: one can get away with $O(\log \log m)$ bits. In this exercise, you need to analyze the following algorithm.

```
1: procedure PROBABILISTIC COUNTER:
2:   Maintain a counter  $C$  initialized to  $C \leftarrow 0$ .
3:   for when element  $e$  arrives do:
4:     With probability  $\frac{1}{2^C}$ , increment  $C \leftarrow C + 1$ .
5:   return  $\text{est} \leftarrow 2^C - 1$ .
```

- Show that est is a unbiased estimate of the number of elements in the stream, that is, $\mathbf{Exp}[\text{est}] = m$.
- Show that the variance of the estimate is $\mathbf{Var}[\text{est}] = \frac{m(m-1)}{2}$.
- For an integer $\alpha \geq 1$, what is the probability that $C \geq \log_2 m + \alpha$? What is the probability that the space required by the above algorithm is $> \log_2 \log_2 m + 1$ bits?
- How many bits are required to obtain $(1 + \varepsilon)$ -approximation to m with probability $\geq 1 - \delta$?
- What happens if we change the probability $\frac{1}{2^C}$ to $\frac{1}{(1+\eta)^C}$ for some $\eta \in (0, 1)$? Can you show how this leads to a much better space bound for a suitable η ?

Problem 26 (Sampling among the distinct elements).

RESERVOIR SAMPLING samples an element i in a stream with probability proportional to f_i . What if we wanted to obtain an element uniformly at random among the distinct elements. That is, we want i to be sampled with probability $\frac{1}{F_0}$ if i is in the stream, and 0 otherwise.

Show that the algorithms done in class can do the job “approximately”. Indeed, show that FLAJOLET-MARTIN can be used to return i in the stream with probability $\geq \frac{1}{cF_0}$ for some constant c . What about BJKST?

Problem 27 (Single Element Recovery). ♥

Suppose $x \in \{0, 1\}^n$ is an **unknown** Boolean vector. The only way you can access it is via a *sum-query*, that is, you can pick a subset $S \subseteq [n]$ and obtain the sum $x(S) := \sum_{i \in S} x_i$. Your goal is to find any one coordinate $i \in [n]$ such that $x_i = 1$, or assert that x is the all zero vector.

- First show that this problem can be solved by a **deterministic** algorithm using $\leq \lceil \log_2 n \rceil + 1$ sum-queries.

The above algorithm however proceeds in rounds. We want an algorithm which makes all questions in a single round. In other words, we want to find a collection of subsets S_1, \dots, S_L such that for any vector $x \in \{0, 1\}^n$, upon receiving the answers $x(S_1), x(S_2), \dots, x(S_L)$, the algorithm can output a coordinate i with $x_i = 1$ with probability $\geq 1 - \delta$.

- b. Suppose you knew that $\sum_{i=1}^n x_i$ is exactly 1. Show there exists a collection of $\lceil \log_2 n \rceil$ sets $A_1, \dots, A_{\lceil \log_2 n \rceil}$ such that given answers $x(A_t)$'s, one can exactly figure out the single coordinate i with $x_i = 1$.
- c. Now suppose you knew $\sum_{i=1}^n x_i = d$. Let R be a subset of $\{1, 2, \dots, n\}$ where each $i \in R$ with probability $\frac{1}{d}$. What is the probability $\Pr[\sum_{i \in R} x_i = 1]$? Furthermore, if it is 1, then can you use the algorithm in part (b) **as a black box** to obtain an element with $x_i = 1$. Argue that $O(\log n \log(1/\delta))$ queries would suffice to get you what you want with probability $\geq 1 - \delta$.
- d. How does your answer in part (c) change if $d \leq \sum_{i=1}^n x_i \leq 2d$?
- e. You don't know d up-front (you could find it out using a single sum-query, but you don't have rounds). Show how you can use $d = 2^i$ for $1 \leq i \leq \lceil \log_2 n \rceil$ to solve the single element recovery problem with $O(\log^2 n \log(1/\delta))$ queries.

Week 8 and 9

Problem 28 (A better deterministic algorithm).

Consider the modification of WEIGHTED MAJORITY where we update the mistaken expert's weights as

$$w_i(t+1) = w_i(t) \cdot (1 - \eta)$$

for some parameter $\eta \in (0, 1/2)$. Prove that if there exists an expert which makes at most k mistakes, then the WEIGHTED MAJORITY algorithm with any parameter $0 \leq \eta \leq \frac{1}{2}$ makes at most $2(1 + \eta)k + \frac{2 \ln n}{\eta}$ mistakes.

Problem 29 (Maximum Coverage).

In class, we looked at the set cover problem which asked the question of the smallest number of sets that need to be picked to cover all elements. In this exercise, we look at a dual problem : given a parameter k , figure out how to pick k subsets so that the maximum number of elements in U are covered. To this end, consider the following LP

$$\text{LP} := \max \sum_{e \in U} z_e \quad \text{s.t.} \quad z_e \leq \sum_{j: e \in S_j} x_j, \quad \sum_{j=1}^m x_j = k, \quad x \in [0, 1]^m, z \in [0, 1]^n$$

x_j is supposed to indicate if S_j is picked. z_e is supposed to indicate if e is covered by the sets picked. We want to maximize the numbers of elements covered. LP, therefore, is an **upper bound** on the maximum elements any k -sets can cover. Can you use the (x, z) to design an algorithm which picks exactly k sets and covers $\geq (1 - \frac{1}{e}) \cdot \text{LP}$ many elements?

You may find the following inequality handy: for any $0 \leq t \leq 1$, $1 - e^{-t} \geq (1 - \frac{1}{e})t$. This itself follows from the "concavity" of the function $1 - e^{-z}$.

Problem 30 (1 Dimensional k -means).

Imagine all the n points P given to you are on a single line. You may assume this line is the x -axis. Now you run the k -means++ algorithm on these set of points. What can you say about the algorithm? First, can you come up with an example where the solution returned is *not* the optimal solution (with $> 1/2$ probability)? Can you prove the approximation factor is better than $O(\ln k)$?