

COS 330: Great Ideas in Theoretical Computer Science

Fall 2026

Problem Set 3

Module: Randomization

Below is a reminder of key aspects of the PSet:

- **The only goal of this PSet is to help you develop your problem-solving skills.** Your performance on this PSet will not directly contribute to your grade, but will indirectly improve your ability to do well on the exams.
 - **Your performance does not directly impact your grade**, so you may use any resources you like (collaboration, AI, etc.) to help you complete the PSet.
 - **AI is not equally welcome in every part of this PSet**, and we say so explicitly where it matters. Problem 1 is exam-style and should be attempted alone first. Problem 2 explicitly calls for AI assistance in part (e). Problem 3 is an optional challenge problem that was designed to be solved without AI assistance.
 - **None of this is an integrity rule** – PSets are not graded directly. It's advice about what will actually develop your skills and in the process prepare you well for the exams, which are closed-book and closed-AI.
 - **See the course website** for a full walkthrough of AI tools and recommendations, as well as the input files needed for Problem 2.
 - **An AI-helper is available for Problem 1 and Problem 2(a)–(d)** – unlike a general assistant, it gives hints and pushes back instead of handing you the answer. See the link above for more information.
 - Throughout the PSet, we've included some general tips to help put these into broader context. Exams will not have these, and future PSets may have fewer.
-

Aligning Expectations

The  symbol implies that the following problem is an “exam-style” problem. We highly recommend that you write up a full solution to this problem as if you were on an exam.

Problem 1 : Meeting in the Middle

There are n people located at coordinates $x_1, \dots, x_n \in \mathbb{R}$ on a line, where $n \geq 3$ is odd. We want to choose a meeting location $y \in \mathbb{R}$ minimizing the total distance everyone travels:

$$F(y) = \sum_{i=1}^n |x_i - y|, \quad \text{OPT} = \min_{y \in \mathbb{R}} F(y).$$

Coordinates need not be distinct and may be arbitrarily large. The median of an odd-length list is its middle entry after sorting, counting repeated entries separately. Throughout, let m be the median of $x_1, \dots, x_n$. Assume arithmetic and comparisons take constant time.

(a) Prove that $F(m) = \text{OPT}$.

(b) Now, suppose each person i reports a coordinate r_i , which may differ from their true location x_i . Then, the process of choosing a meeting location is a mechanism that takes the reports as input and returns a meeting location. The mechanism sees the reports, but person i 's travel distance is always measured from x_i , not r_i .

Fix an odd positive integer s and a list of indices $j_1, \dots, j_s \in \{1, \dots, n\}$, chosen independently of the reports. The mechanism chooses the meeting location to be the median of

$$r_{j_1}, \dots, r_{j_s}.$$

Repeated indices are allowed: if person i appears several times, their single report r_i fills all of those positions. Changing that report changes all its copies together. A person whose index is absent has no entry in the list.

Prove that this mechanism is strategyproof. More precisely, fix any person i , any index list, and any reports from the other people (which need not be truthful). Let y be the meeting location when i reports x_i , and let y' be the meeting location when only i 's report changes to any alternative coordinate. Show that

$$|x_i - y| \leq |x_i - y'|.$$

In other words, lying cannot reduce that person's travel distance. Your proof should apply to every fixed index list, including the full list $1, \dots, n$.

(c) Fix $0 < \alpha < 1/2$. Suppose a location y has at most $(1/2 + \alpha)n$ coordinates strictly to its left and at most $(1/2 + \alpha)n$ coordinates strictly to its right. Prove that

$$F(y) \leq \frac{1/2 + \alpha}{1/2 - \alpha} \text{OPT}.$$

Small Hint

By symmetry, it suffices to consider $y \geq m$. Let $d = y - m$: try bounding $F(y) - F(m)$ from above and $F(m)$ from below in terms of d . Which people's distances decrease by exactly d when the meeting moves from m to y ?

(d) Suppose everyone reports truthfully. Choose an odd number s of indices independently and uniformly from $\{1, \dots, n\}$, with replacement, and return the median Y of the sampled coordinates.

For $0 < \varepsilon \leq 1$ and $0 < \delta \leq 1/4$, show that choosing

$$s = O\left(\frac{1}{\varepsilon^2} \ln \frac{1}{\delta}\right)$$

suffices to guarantee $\Pr[F(Y) \leq (1 + \varepsilon)\text{OPT}] \geq 1 - \delta$. Give an explicit sufficient choice of s and analyze the algorithm's running time in terms of s , assuming a uniform random index and access to its coordinate each take $O(1)$ time. You do not need to compute $F(Y)$.

Small Hint

Use part (c) to decide which meeting locations are good enough. If the sample median is too far to the left, at least half the sampled people must come from a group on the left that contains less than half of the whole population. The same idea applies on the right. How likely is it that such a group makes up half the sample? Count how many sampled people belong to it, and use Hoeffding's inequality to bound the chance that this count is much larger than its expectation. This is a concentration inequality; you can find its statement in the Lecture 6 notes.

This is the most challenging part of Problem 1. Spend at least 15 minutes thinking about it on your own before asking for help. If you are still stuck, use the AI-helper, ask on Ed, or come to office hours.

What We Constructed

With truthful reports, we can choose a meeting location whose total travel distance is at most $(1 + \varepsilon)\text{OPT}$ with probability at least $1 - \delta$, using only $O(\varepsilon^{-2} \ln(1/\delta))$ sampled reports. The number of reports we inspect does not depend on the population size. So, much like the polling example from lecture, we obtain a highly accurate estimate of some quantity (in this case, the optimal meeting location) by looking at only a constant part of the input.

The randomized mechanism is also strategyproof, even though part (d) did not explicitly ask you to prove this. The sample is chosen independently of the reports, and part (b) applies to every fixed sample, including repeated indices. Thus, for every possible sample, no person can reduce their travel distance by lying.

Problem 2 : Selective Memory

Imagine a monitoring system that takes a measurement every second while a machine is running. Each run produces a long list of numbers. Later, we might want to compare a new run with an older one to see whether the machine behaved similarly, or find which past runs most resemble each other.

Keeping every measurement from every run is expensive. Instead, we would like to keep a short summary of each run, discard the original measurements, and still be able to estimate how different two runs were. We do not know in advance which runs will be compared. How much information should a summary keep, and how accurate can the resulting comparisons be?

We represent a run by a record $x = (x_1, \dots, x_d)$: an array of $d \geq 2$ real numbers, where x_i is the measurement at position i . Such an array is also a vector, and so we will also use the notation $x \in \mathbb{R}^d$ to represent it. All records being compared have the same length, with matching positions representing matching measurements. We measure their difference by the squared distance

$$D(x, y) = \sum_{i=1}^d (x_i - y_i)^2.$$

This adds up the squared differences at corresponding positions; it is zero exactly when the two records are identical. An encoder turns a record x into a shorter array $C(x)$, called its message. A decoder takes two messages $C(x)$ and $C(y)$ and estimates $D(x, y)$ without access to the original records.

We will investigate an encoder built from coin flips. Independently choose signs $\sigma_1, \dots, \sigma_d$, each equal to $+1$ or -1 with probability $1/2$. These signs define the function

$$S(x) = \sum_{i=1}^d \sigma_i x_i,$$

which gives our first encoder, $C(x) = (S(x))$: a message containing one number. We will improve this encoder in the following parts. For example, choosing alternating signs for $x = (3, 1, 4, 2)$ gives $S(x) = 3 - 1 + 4 - 2 = 4$.

(a) For two fixed records x, y , apply the function S to both records and consider the estimate

$$Z = (S(x) - S(y))^2.$$

Prove that $\mathbb{E}[Z] = D(x, y)$, where the expectation is over the random signs. Then give two records with $D(x, y) > 0$ for which every possible choice of signs gives

$$|Z - D(x, y)| \geq \frac{1}{2} D(x, y).$$

Thus, an estimate can be correct in expectation and still be unreliable on every individual comparison.

Small Hint

Expand the square and consider separately terms involving one sign and terms involving two different signs. For the example, try two records that differ in just two coordinates.

(b) You may use the following fact without proof: the estimate Z in part (a) satisfies

$$\text{Var}(Z) \leq 2D(x, y)^2.$$

We first improve the encoder by storing several signed sums. For $0 < \varepsilon \leq 1$, let $r = \lceil 8/\varepsilon^2 \rceil$. Choose r independent functions $S_1, \dots, S_r$ of the form defined above: for each S_ℓ , independently choose its d signs once. The new encoder stores

$$C(x) = (S_1(x), \dots, S_r(x)).$$

Use these chosen functions for every record. In particular, entry ℓ in $C(x)$ and entry ℓ in $C(y)$ are computed using the signs of S_ℓ .

The decoder squares the difference between corresponding entries of the two messages and averages these squared differences:

$$Z_\ell = (S_\ell(x) - S_\ell(y))^2, \quad A = \frac{1}{r} \sum_{\ell=1}^r Z_\ell.$$

For any two fixed records x, y , prove that

$$\Pr[|A - D(x, y)| \leq \varepsilon D(x, y)] \geq \frac{3}{4}.$$

Your argument should also cover $D(x, y) = 0$.

Small Hint

Use part (a) to find the average's expectation. How does independence affect its variance? Then apply Chebyshev's inequality. Treat $D(x, y) = 0$ separately.

(c) We now improve the encoder so that its decoder is more reliable. Keep r from part (b). For $0 < \delta \leq 1/4$, let g be the smallest odd integer at least $8 \ln(2/\delta)$. Choose g independent groups of r functions each, denoted $S_{j,1}, \dots, S_{j,r}$ for group j . All rg functions have independently chosen signs. The new encoder concatenates the groups' messages into one array:

$$C(x) = (S_{1,1}(x), \dots, S_{1,r}(x), \dots, S_{g,1}(x), \dots, S_{g,r}(x)).$$

As before, use these chosen functions for every record. For each group j , the decoder applies part (b)'s averaging rule to that group's entries in $C(x)$ and $C(y)$, obtaining

$$A_j = \frac{1}{r} \sum_{\ell=1}^r (S_{j,\ell}(x) - S_{j,\ell}(y))^2.$$

It returns M , the median of $A_1, \dots, A_g$. For any two fixed records x, y , prove that

$$\Pr[|M - D(x, y)| \leq \varepsilon D(x, y)] \geq 1 - \delta.$$

Small Hint

Call a group bad if its estimate is outside the desired interval. How many bad groups are needed for the median to be outside that interval? Apply Hoeffding's inequality to the sum of their indicator variables. This is a concentration inequality; its statement is in the Lecture 6 notes.

(d) So far, we have guaranteed accuracy for one fixed pair of records. We now want the same encoder to give accurate estimates for every pair in a collection of $N \geq 2$ records. The entire collection is fixed before we choose the encoder's random signs. Fix $0 < \varepsilon \leq 1$ and $0 < \delta \leq 1/4$, and let

$$P = \binom{N}{2}, \quad r = \lceil 8/\varepsilon^2 \rceil,$$

with g the smallest odd integer at least $8 \ln(2P/\delta)$. Use exactly the encoder and decoder from part (c), with these values of r, g . Choose the rg functions once and use them to encode all N records.

(i) Let $\hat{D}(x, y)$ denote the decoder's output. Prove that, with probability at least $1 - \delta$ over the choice of the encoder, every pair in the collection simultaneously satisfies

$$|\hat{D}(x, y) - D(x, y)| \leq \varepsilon D(x, y).$$

(ii) Show that each message contains $O(\varepsilon^{-2} \ln(N/\delta))$ numbers. Give running-time bounds, in terms of d, r, g , for encoding one record and for decoding one pair of messages.

Small Hint

First bound the failure probability for one fixed pair using part (c). There are P pairs: which bound combines these failure events without assuming they are independent?

Aligning Expectations – Using AI in This Question

Part (e) asks you to use AI assistance for implementation and experiments. Decide what you want to implement and measure, then ask the assistant to help carry out those plans. You are responsible for checking that the code implements your methods.

(e) On the course website you will find input files containing four collections of records. Inputs 1 and 2 have 20 records of length 32 for checking your implementation. Inputs 3 and 4 each have 604 records of length 4096 for evaluating your methods. The input ZIP contains four numbered `.csv` files whose names identify the category and size. Each row is one record, with comma-separated coordinates. For example, with Python and NumPy, load a file using `np.loadtxt(filename, delimiter=',')`.

For each budget $B \in \{16, 64, 256\}$, design an encoder whose message $C(x)$ is an array of at most B numbers. Any indices or other record-specific information you store also count toward this limit. The decoder must estimate the squared distance using only the two messages. You may fix choices such as random signs or block boundaries in advance and use them for all records.

Have an AI assistant implement the method from part (d), along with code to compare its estimates with the true distances and measure running time. Use the supplied small inputs to check that the code follows your method before moving to the large inputs. On small inputs, storing all 32 coordinates when $B \geq 32$ provides a useful correctness check. Choose r, g with g odd and $rg \leq B$; compare at least two choices at each budget, including $g = 1$. These choices need not meet the worst-case bounds from part (d).

Treat this method as your baseline. Inspect the large inputs, using AI to help you measure their properties, and design at least two alternative methods or changes beyond adjusting r, g . Explain what you expect each idea to exploit, then have AI help implement and test it. Your alternatives need not satisfy the guarantee from part (d), and you do not need to beat the baseline on every input.

Design your own experiments to compare accuracy and encoding and decoding time under the same budget. Decide which pairs of records to compare, how to measure error, and how to summarize the results. Explain why your tests are informative and what kinds of failures they might miss. Use the same comparisons for all methods, and repeat randomized methods with several independent choices of randomness. You may use the original records to compute reference distances, but not to help the decoder answer comparisons.

Report what you learned about the inputs, what you tried, how you measured it, and which method you recommend for each budget. Include unsuccessful attempts and explain what your experiments do and do not establish. Write your explanations in your own words.

Problem 3 : Tiny Counting

Optional – Challenge Problem

This problem is optional and considerably harder than Problems 1 and 2 – we don't suggest attempting it just to practice for the exam. It's here for those who want to engage more deeply with the material; if you're considering an IW/thesis in CS theory, it'll give you a taste of what that might be like. Feel free to discuss it with your TA/UCA coach.

Do not use AI to solve this problem. This is a challenge problem, so if you want to work on it you should do so on your own.

Consider the task of maintaining a counter, which is initially 0. The counter supports two operations:

- `increment()`: Increases the value of the counter by 1.
- `get()`: Returns the current value of the counter.

A simple way to implement this counter is to use a single integer variable to store the current value of the counter. This needs $\lceil \log_2(m+1) \rceil$ bits to count up to m . Here is an alternative way:

- Let X be a variable which is initially 0.

- Implement `increment()` as follows: with probability $1/2^X$, increase X by 1, otherwise, do nothing.
- Implement `get()` as follows: return $2^X - 1$.

(a) Show that after m calls to `increment()`, the expected value returned by `get()` is m and its variance is at most $2m^2 + 1$.

Hints

Let X_m be the state after m increments. Try induction on m , tracking both $\mathbb{E}[2^{X_m}]$ and $\mathbb{E}[2^{2X_m}]$. You can condition on the previous state, or define $P_{m,k} = \Pr[X_m = k]$ and write a recurrence for these probabilities. Remember to subtract the square of the mean when converting the second moment into a variance.

(b) Part (a) shows that one counter gives the correct answer in expectation, but this alone does not ensure that an individual estimate is close to the true count. We will investigate whether averaging several independent estimates gives a reliable answer. Fix $0 < \varepsilon \leq 1$, and maintain $r = \lceil 12/\varepsilon^2 \rceil$ independent copies of the original counter. On each call to `increment()`, update every copy using its own independent randomness. After m calls, let their states be $X_m^{(1)}, \dots, X_m^{(r)}$, and return

$$A = \frac{1}{r} \sum_{\ell=1}^r (2^{X_m^{(\ell)}} - 1).$$

For every fixed $m \geq 1$, prove the following:

- The estimate satisfies $\Pr[|A - m| \leq \varepsilon m] \geq 3/4$.
- One counter's state X_m uses $O(\log \log(m+2))$ bits in expectation, so all r states together use $O(\varepsilon^{-2} \log \log(m+2))$ bits in expectation. Store each state in binary, using one bit for zero.

Count only the stored counter states, not temporary work needed to compute or return the estimate. The algorithm does not know m .

Hints

For accuracy, use part (a) to find the mean and variance of A , then apply Chebyshev's inequality.

For space, the state is X_m , not $2^{X_m} - 1$. A nonnegative integer x needs at most $1 + \log_2(1+x)$ bits. You may use the consequence of concavity that $\mathbb{E}[\log_2 W] \leq \log_2 \mathbb{E}[W]$ for any positive random variable W with finite expectation. Apply this first to $W = 2^{X_m}$ and then to $W = 1 + X_m$.

(c) We now make the estimate more reliable, using the same idea as Problem 2(c). Fix $0 < \delta \leq 1/4$, keep r from part (b), and let g be the smallest odd integer at least $8 \ln(2/\delta)$. Maintain g independent groups of r counters each, updating all rg counters on every call to `increment()`.

To answer `get()`, compute the average from part (b) within each group, obtaining $A_1, \dots, A_g$. Return their median $\hat{m}$. Prove that, after any fixed $m \geq 1$ increments,

$$\Pr[|\hat{m} - m| \leq \varepsilon m] \geq 1 - \delta,$$

and that the expected total number of bits used to store the counter states is

$$O\left(\frac{1}{\varepsilon^2} \ln \frac{1}{\delta} \cdot \log \log(m + 2)\right).$$

As in part (b), count only the stored states. The probability guarantee is for each fixed query time, not simultaneously for all query times. Neither the update rule nor the query rule uses the unknown value m .

Small Hint

Part (b) says that each group gives a good estimate with probability at least $3/4$. If more than half the groups give good estimates, where must their median lie? You may reuse the argument from Problem 2(c), which applies Hoeffding's concentration inequality to the number of bad groups. Its statement is in the Lecture 6 notes. For space, add the expected storage of the g groups.