Homework 6

CSCI 4030 / CSCI 6220: Randomized Algorithms — Fall 2026

Rensselaer Polytechnic Institute — 100 points

Justify every claim. Acknowledge any collaborators and any external sources you consult, including AI tools, and describe clearly what role they played in your submission.

Throughout, log denotes the natural logarithm and $\log_2$ denotes the base-2 logarithm. Ignore integer rounding when it does not affect the substance of an argument.

All three problems of this homework ask you to show the existence of, or construct, a useful combinatorial object that is easy to find using randomness: a cheap sensor layout, a small pooled-testing design, and a short positioning code.

Problem 1 in particular introduces Raghavan's *pessimistic estimators*, a more flexible form of the method of conditional expectations.

[40 points] Problem 1 (Placing sensors).

The problem. Albany wants to install sensors to monitor m regions, choosing from n candidate locations. For each region j, let $A_j \subseteq [n]$ be the locations from which a sensor can monitor it; a region is monitored as soon as a sensor is installed at *one* location in A_j. Every region can be seen from many places: $|A_j| \geq k \geq 1$ for all j.

Sensors are not free, and the cost is borne locally. The locations are partitioned into L maintenance districts, installing at location i costs $c_i \in [0, 1]$, and each district has its own budget. Write C_ℓ for the cost of installing at *every* location of district ℓ. Figure 1 shows a small instance.

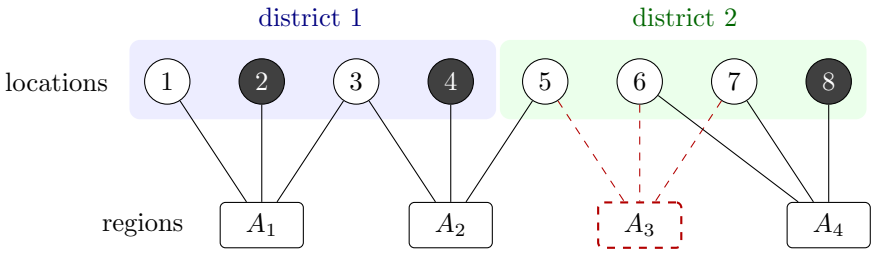


Figure 1: $n = 8$ locations in $L = 2$ districts, $m = 4$ regions, $k = 3$. Edges join each region to the locations that can see it. Sensors at $2, 4, 8$ (filled) monitor A_1, A_2, A_4 but leave A_3 unmonitored. Costs are not shown.

Random placement is a good start. If each location received a sensor independently with probability $1/k$, each region would expect at least one sensor, but "one in expectation" still leaves a constant chance of none. Raising the probability to

$$p = \frac{\log(4m)}{k}$$

makes each region expect $\log(4m)$ sensors, and the chance it gets none drops to $1/(4m)$, small enough to take a union bound over all m regions. Each district, meanwhile, expects to spend only pC_ℓ. These bounds will show that a random placement meets all requirements with positive probability. Our target is to spend at most

$$B_\ell = (e - 1)\, pC_\ell + \log(4L)$$

in each district ℓ: a constant factor above the expected spend, plus a small additive slack. We want a *deterministic* polynomial-time algorithm that monitors every region and keeps every district within budget.

In lecture, the method of conditional expectations fixed the random choices one at a time, each time keeping a conditional expectation from getting worse. Here success means that $m + L$ requirements hold *simultaneously*, and part (b) explains why tracking the exact success probability is unattractive. Instead, we use a *pessimistic estimator*: a tractable upper bound on the indicator of failure whose conditional expectation we can compute efficiently.

The random experiment. Assume that $k \geq \log(4m)$, so that $p \leq 1$. Independently place a sensor at each location with probability p. Write $Y_i = 1$ if location i receives a sensor and $Y_i = 0$ otherwise. Let U be the number of regions left unmonitored, and let $X_\ell = \sum_{i \in \text{district } \ell} c_i Y_i$ be the amount spent in district ℓ.

(a) (10 points) *The requirements can be satisfied.*

(i) Show that region j is unmonitored with probability at most $1/(4m)$, and conclude that $\mathbb{E}[U] \leq 1/4$. (Use $1 - x \leq e^{-x}$.)

(ii) Establish that $\mathbb{P}[X_\ell \geq B_\ell] \leq e^{-B_\ell}\mathbb{E}[e^{X_\ell}]$. Use the inequality $e^c - 1 \leq (e-1)c$ for $c \in [0,1]$ (why does it hold?), to show $\mathbb{E}[e^{X_\ell}] \leq e^{(e-1)pC_\ell}$. Conclude that district ℓ exceeds its budget with probability at most $1/(4L)$.

(iii) Explain why a placement meeting all $m + L$ requirements exists.

(b) (4 points) *Why not track the exact success probability?*

(i) A direct approach would track the conditional probability of success, $\mathbb{P}[U = 0 \text{ and } \forall \ell : X_\ell < B_\ell \mid Y_1 = y_1, \ldots, Y_t = y_t]$. To see a computational obstacle, consider instead one district whose undecided locations are independently selected with probability $1/2$. Assume its costs and budget are rational. Express its budget-violation probability as the number of subsets satisfying a knapsack inequality, divided by a power of two. Explain why this makes directly computing conditional success probabilities unattractive.

(c) (14 points) *The pessimistic estimator.* Combine all the bad events, as in the union bound of part (a), into

$$\Phi = U + \sum_{\ell=1}^{L} e^{X_\ell - B_\ell}.$$

(i) Show that $\Phi \geq 1$ whenever some region is unmonitored *or* some district is over budget. Thus Φ is an upper bound on the indicator of failure; this is why it is called *pessimistic*.

(ii) Show that $\mathbb{E}[\Phi] \leq 1/2$.

(iii) Suppose the first t decisions $Y_1 = y_1, \ldots, Y_t = y_t$ have been made, and $Y_{t+1}, \ldots, Y_n$ are still independent Bernoulli(p). Derive an explicit expression for $\Phi_t(y_1, \ldots, y_t) = \mathbb{E}[\Phi \mid Y_1 = y_1, \ldots, Y_t = y_t]$. Treat separately the regions already monitored by one of locations $1, \ldots, t$ and those not yet monitored, and write each district's term as a product over its undecided locations.

(iv) Show that $\Phi_{t-1}(y_1, \ldots, y_{t-1}) = (1-p)\,\Phi_t(y_1, \ldots, y_{t-1}, 0) + p\,\Phi_t(y_1, \ldots, y_{t-1}, 1)$.

Part (c)(iv) says that, at each step, at least one of the two choices does not increase the estimator. Algorithm 1 always takes such a choice.

Algorithm 1 Sensor placement with a pessimistic estimator

Input: sets $A_1, \dots, A_m \subseteq [n]$ with $|A_j| \geq k \geq \log(4m)$; a district and a cost $c_i \in [0,1]$ for each location i

Output: $y \in \{0,1\}^n$ (install a sensor at location i iff $y_i = 1$)

$p \leftarrow \log(4m)/k$
$B_\ell \leftarrow (e-1)pC_\ell + \log(4L)$ for each district ℓ
for $t = 1, \dots, n$ **do**
    evaluate $\Phi_t(y_1, \dots, y_{t-1}, 0)$ and $\Phi_t(y_1, \dots, y_{t-1}, 1)$ ▷ formula from part (c)(iii)
    $y_t \leftarrow$ the choice in $\{0,1\}$ giving the smaller value (ties: 0)
end for
return y

(d) (8 points) *Correctness.*

(i) Prove the invariant $\Phi_t(y_1, \dots, y_t) \leq 1/2$ for $t = 0, 1, \dots, n$, and conclude that Algorithm 1 monitors every region and keeps every district within its budget.

(ii) Bound the running time of Algorithm 1.

(e) (4 points) *An alternative approach.* Assume $C_\ell > 0$ for every district in this part. Replace each budget term by its Markov version, obtaining $\Phi' = U + \sum_\ell X_\ell / B'_\ell$. Show that Φ' also satisfies (c)(i) and has easily computed conditional expectations. Suppose every budget is the same multiple of the expected spend, $B'_\ell = \beta\, pC_\ell$. Show that guaranteeing $\mathbb{E}[\Phi'] < 1$ requires $\beta > 4L/3$, and compare with B_ℓ. In one or two sentences, say when the simpler Markov estimator is good enough and when the exponential one is worth its extra complexity.

Remark. Sensor placement is a special case of the set cover problem, which we will meet again later in the course.

[30 points] Problem 2 (Identifying infections with pooled COVID tests).

The problem. A laboratory has swab samples from n people, at most d of whom have COVID. Instead of testing each sample separately, the lab splits each sample into portions and mixes portions from different people into a single pool. A pooled test is positive if at least one person in the pool is infected; we assume results are error-free, even for large pools. The lab must choose all pools *before* seeing any results, so that the tests run in parallel, and then identify exactly who is infected. With $n = 10{,}000$ people and $d = 10$, testing individually takes 10,000 tests. How many does pooling need?

Represent t pooled tests by a $t \times n$ binary matrix M: entry $M_{ri} = 1$ means pool r contains a portion of person i's sample. If the infected set is $D \subseteq [n]$, test r is positive exactly when row r has a 1 in some column of D (Figure 2). Assume throughout that $1 \le d < n$.

person	1	2	3	4	5	6	7	result
test 1	1	1	0	0	1	0	0	+
test 2	0	1	1	0	0	1	0	+
test 3	1	0	0	1	0	0	1	−
test 4	0	0	1	1	1	0	0	+
test 5	0	1	0	0	0	1	1	+

Figure 2: Five pooled tests on seven people, with infected set $D = \{2, 5\}$ (shaded). A test is positive iff its row has a 1 in a shaded column. Only test 3 is negative.

Notice that a positive test does not establish that every person in its pool is infected.

We need a matrix in which every person can be told apart from every small group of others. Writing one down explicitly takes real algebraic machinery, but a random sparse matrix does the job, and you will find the right density. The more interesting point comes in part (c). A design that must work for *every* possible infected set is paying for sets that will almost certainly never occur. If the lab instead draws a fresh random design, independently of who happens to be infected, it can get away with far fewer tests.

(a) (8 points) *Decoding.* Say a row r *separates* person i from a set S of other people if $M_{ri} = 1$ and $M_{rj} = 0$ for every $j \in S$. Say M is *d-disjunct* if for every person i and every set S of at most d others, some row separates i from S. Give a simple rule that decides, from the test results alone, whether person i is declared infected, and prove that it recovers every infected set D with $|D| \le d$ when M is d-disjunct. Then apply your rule to Figure 2. Which people does it declare infected, and which separation requirement fails for each mistake?

(b) (12 points) *A design that identifies every infected group.* Let M be a random $t \times n$ matrix whose entries are independently 1 with probability p. Show that if

$$t \ge e(d+1)\big((d+1)\log n + \log 2\big),$$

then M is d-disjunct with probability at least $1/2$, so that $O(d^2 \log n)$ tests suffice. Start by computing the probability that no row separates a fixed person i from a fixed set S with

$|S| = d$, and choose p to minimize it. Explain intuitively why the best pools contain about a $1/(d+1)$ fraction of the people. You may use $(1 - 1/(d+1))^d \geq e^{-1}$, and note that it suffices to consider sets with exactly d people.

(c) (8 points) *A design that identifies one fixed infected group.* Now fix the infected set D, with $|D| \leq d$, and let the lab draw M as in (b), with $p = 1/(d+1)$, independently of D. Show that your decoder outputs exactly D unless some healthy person i has no row separating i from D. Deduce that

$$t \geq e(d+1)\log(n/\delta)$$

tests make the decoder correct with probability at least $1 - \delta$.

(d) (2 points) *Which design to choose?* For $n = 10{,}000$, $d = 10$ and $\delta = 0.01$, compute the two test counts from (b) and (c). In two or three sentences, explain the difference between a verified design that works for *every* infected set and a fresh random design that succeeds with high probability for each *fixed* infected set. When might a lab prefer the former? (Consider a lab that reuses one published design every week.)

Remarks. The extra factor of d in (b) is nearly unavoidable: classical lower bounds for d-disjunct matrices are of order $\min\{n, d^2 \log n / \log d\}$ (D'yachkov and Rykov). Deterministic constructions with $O(d^2 \log n)$ rows are also known (Porat and Rothschild, 2011).

[30 points] Problem 3 (Position labeling).

The problem. Many machines must know exactly where a moving part is: the angle of a robot joint, the position of a tool along a rail. An *incremental* encoder counts ticks as a scale slides past a reader. It is cheap, but one missed tick corrupts every later reading, and after a power loss the machine has to move back to a known "home" position before it can work again. An *absolute* encoder instead reads its position directly from the scale. The classical design prints $\lceil \log_2 n \rceil$ parallel tracks, giving the binary (Gray-code) label of each of the n positions, and reads all tracks at once with one head per track (Figure 3, left).

A *single-track* encoder prints just one row of black and white marks and reads k consecutive marks with k heads placed side by side (Figure 3, right). This works exactly when every window of k consecutive marks occurs at most once on the scale, because a lookup table then maps the window that the heads see to the position. One track is easier to manufacture and needs no alignment between tracks. If k can be made close to $\log_2 n$, it also needs no more heads than the multi-track design. The same idea, in two dimensions, lets some digital pens work on ordinary-looking paper printed with a faint dot pattern. A camera in the pen sees a small patch, and because every patch is unique over a very large area, the pen knows where it is on the page.

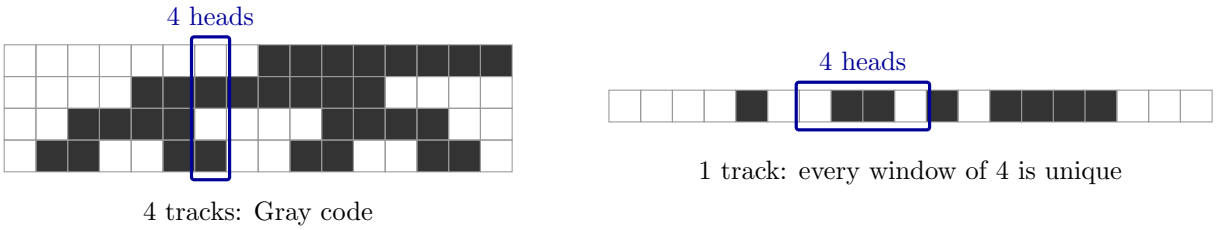


Figure 3: Two absolute encoders for 16 positions. Left: four tracks, read at one position by four heads stacked vertically. Right: a single track (a de Bruijn sequence), read by four heads side by side. The window 0110 under the heads occurs nowhere else on the track, so it identifies the position.

Formally, for $x = x_1 \cdots x_n \in \{0,1\}^n$ let

$$W_i = x_i x_{i+1} \cdots x_{i+k-1}, \qquad 1 \le i \le n-k+1.$$

We want a string whose windows W_i are all distinct, with k as small as possible.

A random string makes any two given windows equal with probability 2^{-k}, so it is natural to try a random scale. For a scale with about a million positions, counting shows that $k \approx 20$ is necessary, but the union bound over all $\approx n^2/2$ pairs of windows only certifies $k \approx 40$. The union bound is pessimistic because it allows all pairs of windows to go wrong together. In fact the event "$W_i = W_j$" depends only on the at most $2k$ bits in those two windows, so most pairs of bad events involve disjoint bits and are independent. The Lovász Local Lemma turns this locality into a much better bound.

(a) (4 points) By counting binary strings of length k, show that $2^k \ge n-k+1$ is necessary.

(b) (8 points) Let $X_1, \ldots, X_n$ be independent uniform random bits. For $i < j$ define the bad event $A_{ij} = \{W_i = W_j\}$. Prove that

$$\mathbb{P}[A_{ij}] = 2^{-k}$$

even when the two windows overlap.

(c) (5 points) Apply the union bound to show that $k \geq 2\log_2 n + O(1)$ is sufficient for a string with no repeated window to exist. Compare with the lower bound from (a).

(d) (7 points) Two bad events are mutually independent whenever they depend on disjoint sets of bit positions, and A_{ij} depends only on the positions in windows i and j. Show that each bad event shares a bit position with at most $d = 4kn$ other bad events. Apply the Lovász Local Lemma to show that, for some absolute contant C, there exists a string with distinct windows of length less than $\log_2 n + \log_2 \log_2 n + C$ for every $n \geq 4$. Thus $k = \log_2 n + \log_2 \log_2 n + O(1)$ suffices, much closer to the counting lower bound than (c).

Algorithm 2 Local Resampling

Input: independent random variables $X_1, \ldots, X_n$; bad events $A_1, \ldots, A_m$, each determined by a set $\mathrm{vbl}(A)$ of the variables

Output: an assignment of the X_ℓ under which no bad event occurs

```
sample every X_ℓ independently
while some bad event A occurs do
    pick any such A (by any fixed rule)
    resample every X_ℓ ∈ vbl(A) independently
end while
return (X_1, ..., X_n)
```

(e) (6 points) *Finding the string.* Algorithm 2 is the local resampling algorithm of Moser and Tardos from lecture. Recall its guarantee: if each of the m bad events has probability at most p, each shares variables with at most d other bad events, and $ep(d+1) \leq 1$, then the algorithm terminates with probability one, and the expected number of resampling steps is at most m/d.

(i) Under the condition you found in (d), show that the expected number of resampling steps is $O(n/k)$.

(ii) Explain how to find a violated event, if one exists, in polynomial time, and bound the expected running time of the algorithm.

Remark. For this exact question there is an explicit optimal answer: a de Bruijn sequence of order k, like the scale in Figure 3, has length $2^k + k - 1$ and contains every k-bit pattern exactly once, so $k = \lceil \log_2 n \rceil$ suffices. The probabilistic argument is still worth knowing because it is robust. Real scales often carry extra requirements: no long runs of one color (for reliable printing), windows that differ in at least two positions (to tolerate a misread mark), or windows that differ from every reversed window (so that the reader can tell which way it is moving). These constraints can also be expressed through bad events involving subsets of the marks. The Local Lemma provides a framework for handling them, provided the event probabilities and dependencies remain sufficiently small. This flexibility is useful even when an optimal explicit construction is available for the basic problem.