# CS49/Math59 Lab 7 Solution Sketches

This writeup provides proof sketches for each problem on Lab 7.

Define the Hamming Distance between two bitstrings as $HD(x,y) := |\{i : x_i \neq y_i\}|$. In other words, $HD(x,y)$ is the number of indices on which $x,y$ differ.

**NOTE:** There are a lot of moving parts in problem (5) and did not expect every student to get it correctly. *Assuming you started this lab early and contacted me if you had questions* consider this lab a success if you had solid solutions for problems 1-4 and made a reasonable attempt at problem (5).

The solution to problem (5) is a good illustration of how error-correcting codes get used in computer science beyond their original motivation.

This lab ties together several recent topics we've discussed in class, including the probabilistic method, communication complexity, randomized algorithms, and error-correcting codes.

Historically, there have been two models of randomized communication: the *public-coin* and the *private-coin* models. In the public-coin model, there is an infinitely long random string $R \in \{0,1\}^*$. Both players see this random string, and can use it to help decide what messages to send. In the private-coin model, Alice and Bob each have their own infinitely-long random string $R_A, R_B : \{0,1\}^*$ respectively. Alice uses $R_A$ and Bob uses $R_B$, and neither player sees the other player's random string (just like they don't see each other's inputs). Randomized protocols are judged by two criteria: *cost* and *error*.

- The *cost* of a protocol $\mathcal{P}$, written cost($\mathcal{P}$), is defined as the worst-case number of bits that get sent during the run of a protocol, taken over all input pairs and all possible choices for the random string(s).
- The *error* of a protocol $\mathcal{P}$, written err($\mathcal{P}$), is defined as:

$$\max_{x,y} \Pr[\mathcal{P}(x,y) \neq f(x,y)] \ .$$

In this lab, you will examine the randomized communication complexity of the EQUALITY function. Recall that EQ : $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is defined as $\text{EQ}(x,y) = 1$ iff $x == y$. This can be modeled as a communication problem by giving Alice $x$ and Bob $y$. We've seen previously that Alice and Bob must exchange at least $n$ bits total to compute EQ using a *deterministic* protocol. Things get easier if we allow *randomized* communication—there is an 8-bit, 1/100-error public-coin protocol for EQ.

1. Consider the following simple two-bit public-coin protocol $\mathcal{P}$ for EQUALITY. Alice and Bob use $R$ to generate a random index $i \in \{1, \ldots, n\}$. Then, they exchange $x_i$ and $y_i$ and output 1 iff $x_i == y_i$. Protocol $\mathcal{P}$ is efficient, but has high error. Compute the error of $\mathcal{P}$.

   **Solution.** First, note that when $x == y$, then $x_i == y_i$ for each $1 \leq i \leq n$. Therefore, Alice/Bob always correctly output 1 when $x, y$ are equal.

   When $x \neq y$, Alice and Bob correctly output 0 if $x_i \neq y_i$. Thus, the *success* probability is equal to $HD(x, y)/n$. Among all $x \neq y$ this is minimized when $HD(x, y) = 1$ i.e. $x, y$ differ in a single bit. In this case, the *success* probability is $1/n$, so $\text{err}(\mathcal{P}) = 1 - 1/n = \frac{n-1}{n}$.

2. Note that the previous protocol has *one-sided error*: if $x == y$, then the protocol always correctly outputs that $x$ and $y$ are equal. Use this observation to create a new protocol $\mathcal{P}'$ with error $\leq 1/4$. What is $\text{cost}(\mathcal{P}')$? (This protocol has good error, but is not terribly communication-efficient.)

   **Solution.** Multiple solutions are possible; here are the two most natural choices:

   - Run the protocol from problem (1) $k$ times, and output 1 iff each run of the protocol outputs 1.
   - Select a set $S \subseteq \{1, \ldots, n\}$ of size $|S| = k$ uniformly at random from all possible such subsets. Then, Alice/Bob exchange $x_i, y_i$ for each $i \in S$ and output 1 iff $x_i == y_i$ for all such $i$.

   These protocols are very similar, in that players exchange $x_i, y_i$ for several randomly sampled indices $i$. In the first protocol, indices are sampled *with replacement.* In the second protocol, they are sampled *without replacement.* Typically, the difference in ampling with or without replacement is negligible, and that's the case here. Now, let's analyze how many bits need to be exchanged for each protocol.

   - In the first case, we only get an error when each of the $k$ instances gives an error. This happens with probability $(1 - 1/n)^k$. We want this error to be at most 1/4. Using $1 + x \leq e^x$ and setting the error to be at most 1/4, we have an error of at most

     $$\left(1 - \frac{1}{n}\right)^k \leq e^{-k/n} \leq 1/4 \ ,$$

     which happens iff $k \geq nln(4) \approx 1.39n$. (Note that this protocol is even worse than if Alice just sent her entire input to Bob!)
   - In the second case, we get a worst-case error if the single $i$ where $x, y$ differ does not get selected. Since we select $k$ out of $n$ coordinates, this happens with probability $1 - k/n$. This is at most 1/4 iff $k \geq 3n/4$.

3. In the lab exercises for week 10, you showed how to take a randomized algorithm $\mathcal{A}$ that takes one of $N$ possible inputs and had 1/4 error and produce another randomized algorithm $\mathcal{A}'$ for the same problem that had 1/3 error and used only $O(\log \log N)$ random bits.

   Use this derandomization result together with the $O(1)$-bit, (1/100)-error public-coin protocol for EQ and show there exists a 1/3-error public-coin EQ protocol that uses only $O(\log n)$ random bits.

**Solution.** The proof of the week 10 lab exercises can go through basically verbatim. The input for the EQUALITY protocol is a pair of $n$-bit strings. Thus, there are $N := 2^n \times 2^n = 2^{2n}$ different possible inputs for the EQUALITY protocol. Hence we can use the derandomization result from week 10 to get the randomness down to $O(\log \log N) = O(\log \log(2^{2n})) = O(\log(n))$ random bits.

4. Using your answer to the previous problem, show that there exists a 1/3-error, private-coin protocol[1] $Q$ for EQUALITY that has $\text{cost}(Q) = O(\log n)$.

   **Solution.** The protocol from problem (3) has 1/3-error and $O(1)$-bits of communication and uses just $O(\log n)$ public random bits. To get a private-coin protocol out of this, Alice generates $O(\log n)$ random bits (using her private randomness) and sends it to Bob, afterwhich they run the protocol from problem (3) using these random bits as the "public" randomness. The error remains 1/3, and the communication cost is now $O(1) + O(\log n) = O(\log n)$.

5. The previous answer showed that it's possible to compute EQUALITY using private randomness and only $O(\log n)$ communication. However, this result has a drawback—the protocol itself is not explicitly constructed. In this problem, your task is to design an *explicit* $O(\log n)$-bit, 1/3-error protocol for EQUALITY.

   **Hint:** First, create an $O(\log n)$-bit, $(1 - 1/320)$-error protocol by using error-correcting codes. Your protocol should have one-sided error, erring only when $x \neq y$. Then, repeat this initial protocol several times to drive down the error. How many times do you need to repeat the protocol to achieve error 1/3?

   **Solution.** This protocol ties together intuition from a lot of different things you've learned about over the past month.

   A first piece of intuition is that the protocol from problem (1) has really terrible error because the input strings $x, y$ could differ in just one coordinate. In problem (2) you saw how to drive this error down to 1/4 (good!) but at a cost of $\Theta(n)$ communication (bad!). The main reason why the blowup in communication was so high was because the error in the protocol from problem (1) was $1 - 1/n$.

   The protocol from problem (1) would run much better if only we could somehow guarantee that the Hamming Distance between the input strings was large. Error correcting codes do just that.

   Recall that we saw in class an ECC based on the magic graph construction that had rate 1/4 and distance $1/10d$, and that we could construct suitable magic graphs as long as $d \geq 32$. Let's fix such an ECC.

   Imagine a world in which the strings for EQUALITY are **codewords** for the ECC. Then, if $x \neq y$, we have $HD(x, y) \geq n/10d$, and the error in the protocol from problem (1) would be $1 - 1/10d$. Make $d$ as small as possible to minimize this error, so we set $d := 32$, and the error is $1 - 1/320 \approx 0.997$. Now, this still seems like a pretty terrible error, but because 0.997 is a

[1] In fact, this result doesn't hold just for EQUALITY. The derandomization result from lab during week 10 actually yields the following result: for any function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, if there is a public-coin, $\varepsilon$-error protocol that uses $t$ bits of communication, then there is a $(\varepsilon + \delta)$-error, **private-coin** protocol for $f$ that uses $t + O(\log(n/\delta))$ bits of communication.

constant *bounded away from 1* we can actually run the repeated protocol (as in problem (2)) to drive down this error efficiently.

Specifically, taking the sampling-with-replacement version, we need $k$ such that $(1-1/320)^k \leq 1/4$. This happens iff $k \geq 443$. Note that $k$ is a constant, whereas in the protocol from problem (2), $k = \Omega(n)$. Thus, getting a reasonable error only costs us a constant blow-up in communication.

There are still two catches with the protocol above. First, the protocol from problem(1) was public-coin, and we need a private coin protocol. Second, we must create a protocol that works for *any* pair of $n$-bit strings $x, y$. The solution is to encode $x, y$ using the error-correcting code, then run a private-coin version of protocol (1), as was done in problem (4). The complete solution is below.

(a) Let $C$ be an error-correcting code with rate $1/4$ and distance $1/320$ as guaranteed by the magic-graph-based ECC we saw in class. Let $ENC$ denote the encoding function. Use a code that maps $n$-bit strings to $n' = 4n$ bit strings. (the rate is $1/4 = n/n'$).

(b) On inputs $x, y \in \{0,1\}^n$, Alice and Bob individually compute $x' := ENC(x)$ and $y' := ENC(y)$.

(c) Then, using her private randomness Alice generates $k = O(1)$ random indices $i_1, \ldots, i_k \in \{1, \ldots, n'\}$. Let $S := \{i_1, \ldots, i_k\}$.

(d) First, she sends $i_1, \ldots, i_k$ to Bob.

(e) Then, she sends $x'_i$ for each $i \in S$ to Bob.

(f) for each $i \in S$, Bob compares $x'_i$ to $y'_i$ and outputs 1 iff each pair of bits is equal.

As with other EQUALITY protocols, this protocol has one-sided error (when $x = y$, the protocol always says so. Because of the guarantee of the error-correcting code, when $x \neq y$, $HD(x', y') \geq n/320$. The protocol produces a false positive only when $x'_i == y'_i$ for each $i$, which happens with probability $(1 - 1/320)^k \leq 1/4$.

The protocol is private coin because Alice generates the randomness herself and communicates it to Bob. The cost of this protocol is $O(k \log(n')) = O(\log n)$ since $k$ is a constant and $n' = O(n)$.