Problem 7.1 (PRGs imply hard functions)

Suppose that for every \( m \), there exists a mildly explicit \((m, 1/m)\) pseudorandom generator \( G_m : \{0,1\}^{d(m)} \to \{0,1\}^m \). Show that \( E \) has a function \( f : \{0,1\}^{\ell} \to \{0,1\}^{\ell'} \) with nonuniform worst-case hardness \( t(\ell) = \Omega(d^{-1}(\ell - 1)) \). In particular, if \( d(m) = O(\log m) \), then \( t(\ell) = 2^{\Omega(\ell)} \) (Hint: look at a prefix of \( G \)'s output.)

Problem 7.14 (PRGs from 1–1 One-Way Functions)

A random variable \( X \) has \((t, \varepsilon)\) pseudoentropy at least \( k \) if it is \((t, \varepsilon)\) indistinguishable from some random variable of min-entropy at least \( k \).

1. Suppose that \( X \) has \((t, \varepsilon)\) pseudoentropy at least \( k \) and that \( \text{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m \) is a \((k, \varepsilon')\)-extractor computable in time \( t' \). Show that \( \text{Ext}(X, U_d) \) is an \((t - t', \varepsilon + \varepsilon')\) indistinguishable from \( U_m \).

2. A \textit{hardcore predicate} for a one-way function \( f : \{0,1\}^{\ell} \to \{0,1\}^{\ell'} \) is a poly\((\ell)\)-time computable function \( b : \{0,1\}^{\ell} \to \{0,1\} \) such that for every constant \( c \), every nonuniform algorithm \( A \) running in time \( \ell^c \), we have:

\[
\Pr[A(f(U_\ell)) = b(U_\ell)] \leq \frac{1}{2} + \frac{1}{\ell^c},
\]

for all sufficiently large \( \ell \).

Let \( f : \{0,1\}^{\ell} \to \{0,1\}^{\ell'} \) be a one-to-one one-way function (not necessarily length-preserving) and \( b : \{0,1\}^{\ell} \to \{0,1\} \) a hardcore predicate for \( f \). Show that for every constant \( c \) and all sufficiently large \( \ell \), the random variable \( f(U_\ell)b(U_\ell) \) has \((\ell^c, 1/\ell^c)\) pseudoentropy at least \( \ell + 1 \).
3. Problems 7.12 and 7.13 show that if $f$ is a one-way function, then $b(x,r) = \langle x, r \rangle \mod 2$ is a hard-core predicate for the one-way function $f'(x,r) = (f(x),r)$, where $|x| = |r|$. Using this result, show how to construct a cryptographic pseudorandom generator from any one-to-one one-way function. (Any seed length $d(m) < m$ is fine.)

**Problem 7.5 (Strong Pseudorandom Generators)**

By analogy with strong extractors, call a function $G : \{0,1\}^d \rightarrow \{0,1\}^m$ a $(t,\varepsilon)$ strong pseudorandom generator if the function $G'(x) = (x,G(x))$ is a $(t,\varepsilon)$ pseudorandom generator.

1. Show that there do not exist strong cryptographic pseudorandom generators.

2. Show that the Nisan–Wigderson generator (Theorem 7.24) is a strong pseudorandom generator.

3. Suppose that for all constants $\alpha > 0$, there is a strong and fully explicit $(m,\varepsilon(m))$ pseudorandom generator $G : \{0,1\}^m \rightarrow \{0,1\}^m$. Show that for every language $L \in \text{BPP}$, there is a deterministic polynomial-time algorithm $A$ such that for all $n$, $\Pr_{x \in \{0,1\}^n}[A(x) \neq \chi L(x)] \leq 1/2^n + \varepsilon(\text{poly}(n))$. That is, we get a polynomial-time average-case derandomization even though the seed length of $G$ is $d(m) = m^\alpha$.

4. (*) Show that for every language $L \in \text{BPAC}^0$, there is an (uniform) $\text{AC}^0$ algorithm $A$ such that $\Pr_{x \in \{0,1\}^n}[A(x) \neq \chi L(x)] \leq 1/n$. You may use the fact that uniform $\text{AC}^0$ algorithms can compute the Parity and Majority functions on polylog($n$) bits. (Warning: be careful about error reduction.)

**Problem 7.6 (Private Information Retrieval)**

The goal of private information retrieval is for a user to be able to retrieve an entry of a remote database in such a way that the server holding the database learns nothing about which database entry was requested. A trivial solution is for the server to send the user the entire database, in which case the user does not need to reveal anything about the entry desired. We are interested in solutions that involve much less communication. One way to achieve this is through replication. Formally, in a $q$-server private information-retrieval (PIR) scheme, an arbitrary database $D \in \{0,1\}^n$ is duplicated at $q$ non-communicating servers. On input an index $i \in [n]$, the user algorithm $U$ tosses some coins $r$ and outputs queries $(x_1, \ldots, x_q) = U(i,r)$, and sends $x_j$ to the $j$th server. The $j$th server algorithm $S_j$ returns an answer $y_j = S_j(x_j, D)$. The user then computes its output $U(i,r,x_1, \ldots, x_q)$, which should equal $D_i$, the $i$th bit of the database. For privacy, we require that the distribution of each query $x_j$ (over the choice of the random coin tosses $r$) is the same regardless of the index $i$ being queried.

It turns out that $q$-query locally decodable codes and $q$-server PIR are essentially equivalent. This equivalence is proven using the notion of smooth codes. A code $\text{Enc} : \{0,1\}^n \rightarrow \Sigma^*$ is a $q$-query smooth code if there is a probabilistic oracle algorithm $\text{Dec}$ such that for every message $x$ and every $i \in [n]$, we have $\Pr[\text{Dec}^{\text{Enc}(x)}(i) = x_i] = 1$ and $\text{Dec}$ makes $q$ nonadaptive queries to its oracle, each of which is uniformly distributed in $[\hat{n}]$. Note that the oracle in this definition is a valid codeword,

\[\text{1}^{\text{Another way is through computational security, where we only require that it be computationally infeasible for the database to learn something about the entry requested.}}\]
with no corruptions. Below you will show that smooth codes imply locally decodable codes and PIR schemes; converses are also known (after making some slight relaxations to the definitions).

1. Show that the decoder for a $q$-query smooth code is also a local $(1/3q)$-decoder for $\text{Enc}$.

2. Show that every $q$-query smooth code $\text{Enc} : \{0, 1\}^n \to \Sigma^\hat{n}$ gives rise to a $q$-server PIR scheme in which the user and servers communicate at most $q \cdot (\log \hat{n} + \log |\Sigma|)$ bits for each database entry requested.

3. Using the Reed-Muller code, show that there is a polylog($n$)-server PIR scheme with communication complexity polylog($n$) for $n$-bit databases. That is, the user and servers communicate at most polylog($n$) bits for each database entry requested. (For constant $q$, the Reed-Muller code with an optimal systematic encoding as in Problem 5.4 yields a $q$-server PIR with communication complexity $O(n^{1/(q-1)})$.)