CS 225: Pseudorandomness	**Prof. Salil Vadhan**
Problem Set 6	
Assigned: Apr. 26, 2007	Due: May. 9, 2007 (1 PM)

- This problem set is a substitute for the final exam. You must work *alone* (but you still may come to office hours with questions about the material!) You are allowed to use up to 7 of your remaining late days (out of the 12 that were initially given to you).
- Recall that your problem set solutions must be typed. You can email your solutions to `cs225-hw@eecs.harvard.edu`, or turn in it to Carol Harlow in MD 343. You may write formulas or diagrams by hand. Aim for clarity and conciseness in your solutions, emphasizing the main ideas over low-level details. If you use LaTeX, please submit both the source (`.tex`) and the compiled file (`.ps` or `.pdf`). Name your files `PS6-yourlastname`.
- Starred problems are extra credit.

Problem 1. (PRGs imply hard functions) Suppose that for every ℓ, there exists a $(t(\ell), 1/t(\ell))$ PRG $G_\ell : \{0,1\}^\ell \to \{0,1\}^{t(\ell)}$ computable in time $2^{O(\ell)}$. (In this problem, it is cleaner to treat the output length as a function of the seed length.) Show that $\mathbf{E}$ has a function $f : \{0,1\}^\ell \to \{0,1\}$ with (nonuniform) worst-case hardness $\Omega(t(\ell-1))$ on inputs of length ℓ. (Hint: look at a prefix of G's output.)

Problem 2. (Deterministic Approximate Counting) Using the PRG for constant-depth circuits (unbounded fan-in) constructed in class, give deterministic quasipolynomial-time algorithms for the problems below. (The running time of your algorithms should be $2^{\mathrm{poly}(\log n, \log(1/\varepsilon))}$, where n is the size of the circuit/formula given and ε is the accuracy parameter mentioned.)

(a). Given a constant-depth circuit C and $\varepsilon > 0$, approximate the fraction of inputs x such that $C(x) = 1$ to within an additive error of ε.

(b). Given a DNF formula φ and $\varepsilon > 0$, approximate the number of assignments x such that $\varphi(x) = 1$ to within a *relative* error of ε. You may restrict your attention to φ in which all clauses contain the same number of literals. (Hint: Study the randomized DNF counting algorithm from Lecture 3.)

Note that these are *not* decision problems, whereas classes such as $\mathbf{BPP}$ and $\mathbf{BPAC_0}$ are classes of decision problems. One of the points of this problem is to show how derandomization can be used for other types of problems.

Problem 3. (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.[1] 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 $U(i, r, y_1, \ldots, y_q)$, 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 $\mathrm{Enc} : \{0,1\}^n \to \Sigma^{\hat{n}}$ is a *q-query smooth code* if there is a probabilistic oracle algorithm Dec such that for every message x and every $i \in [n]$, we have $\Pr[\mathrm{Dec}^{\mathrm{Enc}(x)}(i) = x_i] = 1$ and 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, 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).

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

(b). Show that every q-query smooth code $\mathrm{Enc} : \{0,1\}^n \to \Sigma^{\hat{n}}$ 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.

(c). 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 yields a q-server PIR with communication complexity $O(n^{1/(q-1)})$.)

Problem 4. (Better Local Decoding of Reed–Muller Codes*) Show that for every constant $\varepsilon > 0$, there is a constant $\gamma > 0$ such that there is a local $(1/2 - \varepsilon)$-decoding algorithm for the m-variate Reed–Muller code of degree d over field $\mathbb{F}$ provided $d \leq \gamma|\mathbb{F}|$. The running time should be $\mathrm{poly}(m, |\mathbb{F}|)$.

Problem 5. (Hitting-Set Generators) A set $H_n \subseteq \{0,1\}^n$ is a (t, ε) *hitting set* if for every nonuniform algorithm T running in time t that accepts greater than an ε fraction of n-bit strings, T accepts at least one element of H_n.

(a). Show that if, for every n, we can construct an $(n, 1/2)$ hitting set H_n in time $s(n) \geq n$ where $s(n)$ is nondecreasing, then $\mathbf{RP} \subseteq \mathbf{DTIME}(\bigcup_c n^c \cdot s(n^c))$. In particular, if $s(n) = \mathrm{poly}(n)$, then $\mathbf{RP} = \mathbf{P}$.

(b). Show that if there is a (t, ε) pseudorandom generator $G_n : \{0,1\}^\ell \to \{0,1\}^n$ computable in time s, then there is a (t, ε) hitting set H_n constructible in time $2^\ell \cdot s$.

[1] 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.

(c). Define the notion of a (t, k, ε) black-box construction of hitting set-generators, and show that, when $t = \infty$, such constructions are equivalent to constructions of *dispersers* (as defined in Lecture Notes 11).