CS 225: Pseudorandomness **Prof. Salil Vadhan**

Problem Set 5

Assigned: Apr. 12, 2007 Due: Apr. 25, 2007 (1 PM)

- 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 `PS5-yourlastname`.
- Starred problems are extra credit.

Problem 1. (Concatenated Codes) For codes $\mathrm{Enc}_1 : \{1, \ldots, N\} \to \Sigma_1^{n_1}$ and $\mathrm{Enc}_2 : \Sigma_1 \to \Sigma_2^{n_2}$, their *concatenation* $\mathrm{Enc} : \{1, \ldots, N\} \to \Sigma_2^{n_1 n_2}$ is defined by

$$\mathrm{Enc}(m) = \mathrm{Enc}_2(\mathrm{Enc}_1(m)_1)\mathrm{Enc}_2(\mathrm{Enc}_1(m)_2) \cdots \mathrm{Enc}_2(\mathrm{Enc}_1(m)_{n_1}).$$

This is typically used as a tool for reducing alphabet size, e.g. with $\Sigma_2 = \{0, 1\}$.

(a). Prove that if Enc_1 has minimum distance δ_1 and Enc_2 has minimum distance δ_2, then Enc has minimum distance at least $\delta_1\delta_2$.

(b). Prove that if Enc_1 is $(1 - \varepsilon_1, \ell_1)$ list-decodable and Enc_2 is (δ_2, ℓ_2) list-decodable, then Enc is $((1 - \varepsilon_1\ell_2) \cdot \delta_2, \ell_1\ell_2)$ list-decodable.

(c). By concatenating a Reed–Solomon code and a Hadamard code, show that for every $n \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit code $\mathrm{Enc} : \{0, 1\}^n \to \{0, 1\}^{\hat{n}}$ with blocklength $\hat{n} = O(n^2/\varepsilon^2)$ with minimum distance at least $1/2 - \varepsilon$. (Throughout this problem, measure running time as a function of $\hat{n}$ to handle the case that ε is smaller than $1/\mathrm{poly}(n)$.) Furthermore, show that with blocklength $\hat{n} = \mathrm{poly}(n, 1/\varepsilon)$, we can obtain a code that is $(1/2 - \varepsilon, \mathrm{poly}(1/\varepsilon))$ list-decodable in *polynomial time.* (Hint: the inner code can be decoded by brute force.)

Problem 2. (List-decoding Reed–Solomon Codes) You may ignore round-off errors in your solution to this problem.

(a). Show that there is a polynomial-time algorithm for list-decoding the Reed-Solomon codes of degree d over $\mathbb{F}_q$ up to distance $1 - \sqrt{2d/q}$, improving the $1 - 2\sqrt{d/q}$ bound from lecture. (Hint: do not use fixed upper bounds on the individual degrees of the interpolating polynomial $Q(X, Y)$, but rather allow as many monomials as possible.)

(b). (*) Improve the list-decoding radius further to $1 - \sqrt{d/q}$ by using the 'multiple-roots' trick described in Section 4 of Lecture Notes 15.

Problem 3. (Codes vs. Hashing) Given any code $\mathrm{Enc} : [N] \to [M]^{\hat{n}}$, we can obtain a family of hash functions $\mathcal{H} = \{h_i : [N] \to [M]\}_{i \in [\hat{n}]}$ defined by $h_i(x) = \mathrm{Enc}(x)_i$, and conversely.

(a). Show that Enc has minimum distance at least δ iff $\mathcal{H}$ has collision probability at most $1-\delta$. That is, for all $x \neq y \in [N]$, we have $\Pr_i[h_i(x) = h_i(y)] \leq 1 - \delta$. (This a generalization of the definition of universal hash functions, which correspond to the case that $\delta = 1 - 1/M$.)

(b). It was shown in section that the Leftover Hash Lemma extends to families of functions with low collision probability; specifically if a family $\mathcal{H}$ with range $[M]$ has collision probability at most $(1+\varepsilon^2)/M$, then $\mathrm{Ext}(x,h) = (h, h(x))$ is a (k, ε) extractor for $k = m + 2\log(1/\varepsilon) + O(1)$, where $m = \log M$. Use this to prove the Johnson Bound for small alphabets: if a code $\mathrm{Enc} : [N] \to [M]^{\hat{n}}$ has minimum distance at least $1 - 1/M - \gamma/M$, then it is $(1 - 1/M - \sqrt{\gamma}, O(M/\gamma))$ list-decodable.

Problem 4. (Twenty Questions) In the game of 20 questions, an oracle has an arbitrary secret $s \in \{0,1\}^n$ and the aim is to determine the secret by asking the oracle as few yes/no questions about s as possible. It is easy to see that n questions are necessary and sufficient. Here we consider a variant where the oracle has two secrets $s_1, s_2 \in \{0,1\}^n$, and can adversarially decide to answer each question according to either s_1 or s_2. That is, for a question $f : \{0,1\}^n \to \{0,1\}$, the oracle may answer with either $f(s_1)$ or $f(s_2)$. Here it turns out to be impossible to pin down either of the secrets with certainty, no matter how many questions we ask, but we can hope to compute a small list L of secrets such that $|L \cap \{s_1, s_2\}| \neq \emptyset$. (In fact, $|L|$ can be made as small as 2.) This variant of twenty questions apparently arose from Internet routing algorithms used by Akamai.

(a). Let $\mathrm{Enc} : \{0,1\}^n \to \{0,1\}^{\hat{n}}$ be a code such that that every two codewords in Enc *agree* in at least a $1/2 - \varepsilon$ fraction of positions and that Enc has a polynomial-time $(1/4 + \varepsilon, \ell)$ list-decoding algorithm. Show how to solve the above problem in polynomial time by asking the $\hat{n}$ questions $\{f_i\}$ defined by $f_i(x) = \mathrm{Enc}(x)_i$.

(b). Taking Enc to be the code constructed in Problem 1, deduce that $\hat{n} = \mathrm{poly}(n)$ questions suffices.

Problem 5. (Limitations on the Seed Length) Prove that a cryptographic pseudorandom generator cannot have seed length $\ell(n) = O(\log n)$. Note where your proof fails if we only require that it is an $(n^d, 1/n^d)$ pseudorandom generator for a fixed constant d.