CS 151 Complexity Theory **Spring 2017**

Solution Set 4

Posted: May 3 *Chris Umans*

If you are turning in this problem set late, obviously you shouldn't consult these solutions.

1. Consider a language $L \in \widetilde{\mathbf{ZPP}}$ decided by a machine M that runs in expected time n^k for some constant k. By Markov's inequality, we see that

$$\Pr_y[\#\text{ steps before } M(x,y) \text{ halts} \geq 2|x|^k] \leq 1/2.$$

We build a new machine M' that simulates M and after $2|x|^k$ simulation steps, it outputs "fail". The new machine certainly runs in polynomial time, with probability at most $1/2$ it outputs "fail," and otherwise it accepts or rejects x correctly. Thus $L \in \mathbf{ZPP}$.

In the other direction, consider a language $L \in \mathbf{ZPP}$ with associated probabilistic TM M that runs in time n^k and outputs "fail" with probability $1/2$ and otherwise it accepts or rejects its input correctly. We will simulate M repeatedly until it does not output "fail" (and then accept or reject the input based on M's answer). The expected number or repetitions is

$$\sum_{i=1}^{\infty} 2^{-i} i = \frac{1/2}{(1-(1/2))^2} = 2,$$

and so the expected running time is $2n^k$. Thus $L \in \widetilde{\mathbf{ZPP}}$. We conclude that $\widetilde{\mathbf{ZPP}} = \mathbf{ZPP}$ as required.

2. (a) Since S is non-empty it contains some $j \in \{1, 2, 3, \ldots, \ell\}$. Imagine choosing v_j last. We have $u_S = \alpha$ iff $v_j = \alpha - \sum_{i \in (S - \{j\})} v_i$. This happens with probability exactly 2^{-k}.

Now consider $S \neq T$, and WLOG let j be an element of S that is not in T, and let j' be an element of T (which is nonempty). Imagine choosing all the v_i except v_j and v'_j. There is a unique pair of values for v_j and v'_j that make $u_S = \alpha$ and $u_T = \beta$, and so $\Pr[u_S = \alpha \wedge u_T = \beta] = 2^{-2k}$ as required.

(b) As suggested, we define the indicator random variables X_S for the event $R_{u_S + e_i} = C_{u_S + e_i}$ and $X = \sum_{S \neq \emptyset} X_S$. Note that $\mathrm{E}[X_S] \geq 1/2 + \epsilon$ and $\mathrm{Var}[X_S] \leq 1/4$. Since the $\{U_S\}$ are pairwise-independent, the $\{X_S\}$ are as well. Thus $\mathrm{Var}[X] \leq (2^\ell - 1)/4$ (since the variance of the sum is the sum of the variances for pairwise independent random variables), and $\mathrm{E}[X] \geq (1/2 + \epsilon)(2^\ell - 1)$ (by linearity of expectation). Now we have:

$$\Pr\left[|\{S \neq \emptyset : C_{u_S} + R_{u_S + e_i} = m_i\}| \leq \frac{2^\ell - 1}{2}\right] = \Pr\left[X \leq \frac{2^\ell - 1}{2}\right]$$

$$\leq \Pr[|X - \mathrm{E}[X]| \geq \epsilon(2^\ell - 1)] \leq \frac{\mathrm{Var}[X]}{\epsilon^2(2^\ell - 1)^2}$$
$$\leq \frac{1}{4\epsilon^2(2^\ell - 1)}$$

where the second-to-last line invokes Chebyshev's Inequality.

(c) Our probabilistic procedure A works as follows: set $\ell = \log(k/\epsilon^2 + 1)$, pick ℓ vectors $v_1, v_2, \ldots, v_\ell \in \mathbb{F}_2^k$ independently and uniformly at random, and form the vectors $u_S = \sum_{i \in S} v_i$ as in part (a). Note that $C_v + C_{v'} = C_{v+v'}$. This leads to the key observation: given C_{v_i} for all i, we can produce C_{u_S} for any non-empty S because $C_{u_S} = \sum_{i \in S} C_{v_i}$. We *enumerate* all possible "guesses" for the C_{v_i} – call g_i the guess for C_{v_i}. There are a total of $L = 2^\ell = O(k/\epsilon^2)$ such sets of guesses, and for each we will output a message for our list.

Specifically, for each such vector of guesses $(g_i)_{i=1,2,\ldots,\ell}$ we do the following. For each j and S, we compute $z_{j,S} = (\sum_{i \in S} g_i) + R_{u_S + e_j}$. If the guesses are correct, i.e., $g_i = C_{v_i}$ for all i, then our analysis in part (b) shows that for each j, with probability at least $1 - \frac{1}{4k}$ the majority of the $z_{j,S}$ (over S) give m_j. By a union bound, the probability that $y_j = \mathrm{maj}_S(z_{j,S}) = m_j$ for all j is at least $3/4$. Thus if we output $y_1 y_2 \ldots y_k$ for each guess, we will include the original message m in our list with probability at least $3/4$.

The running time of the above procedure is $\mathrm{poly}(k, \epsilon^{-1})$ and it makes $k(2^\ell - 1) = k^2/\epsilon^2$ queries to the received word R for each set of guesses, for an overall number of queries $\mathrm{poly}(k, \epsilon^{-1})$.

(d) Let us call an x "good" if

$$\Pr_y[C_n(x, y) = GL(f'_n(x, y))] \geq 1/2 + \epsilon(n)/2.$$

Notice that $GL(f'_n(x, y))$ is the y-th bit of the Hadamard encoding $C(f_n(x))$. If x is "good" then $C_n(x, y)$ may be thought of as a received word that agrees with $C(f_n(x))$ on $1/2 + \epsilon(n)/2$ fraction of its positions.

Plugging C_n into our probabilistic procedure A from the previous part (i.e., invoking $C_n(x, y)$ whenever we would have queried the y-th bit of R), we get a probabilistic procedure A' that outputs a list of $L = O(n/\epsilon(n))$ candidate messages, that with probability at least $3/4$ includes $f_n(x)$ (our "message").

By modifying A' one more time so that it outputs a random one of the L messages, we obtain finally a probabilistic procedure that takes as input (x, y), and outputs $(f_n(x), y)$ if x is "good," with probability at least $(3/4)L^{-1} = O(\epsilon(n)^2/n)$. Since x is "good" with probability at least $\epsilon(n)/2$, the overall probability that our probabilistic procedure outputs $f'_n(x, y) = (f_n(x), y)$ is at least $(\epsilon(n)/n)^{O(1)}$ as required.

The running time of the procedure A is $(n/\epsilon)^{O(1)}$ from the previous part, and in the worst case we invoke $C_n(x, y)$ (a size $s(n)$ circuit) at each step to answer a query. Thus the overall running time is at most $(s(n)/\epsilon(n))^{O(1)}$ (using the fact that $s(n) \geq n$). It is easy to convert our current probabilistic procedure into a circuit with at most a quadratic blowup (using the CVAL construction), and then we can fix the random bits to preserve the probability of outputting the correct answer. Altogether, the size of the circuit family derived in this way is $(s(n)/\epsilon(n))^{O(1)}$ as required.

3. (a) A generic polynomial $Q(x, y)$ with x-degree and y-degree at most $\sqrt{q}$ has the form $Q(x, y) = \sum_{m \in M} c_m m(x, y)$ where M is the set of monomials in x, y with the required degree constraints (e.g. $x^2 y$). Note that $|M| = (\sqrt{q} + 1)^2 > q$. We have q constraints of the form $Q(w, R(w)) = 0$ for $w \in \mathbb{F}_q$ that we wish to satisfy. Each of these is a linear constraint on the coefficients c_m of the form: $\sum_{m \in M} c_m m(w, R(w)) = 0$. Thus we have a homogeneous system of q linear equations in $|M| > q$ variables, so there is a non-trivial solution (i.e., one in which not all the c_m are zero). This gives us the desired polynomial $Q(x, y)$, and because solving this system of equations can be done in a straightforward way using Gaussian elimination, Q can be found efficiently.

   (b) Consider $Q'(x) = Q(x, p(x))$. By construction $Q'(x) = Q(x, p(x)) = 0$ for those $x \in \mathbb{F}_q$ on which p and R agree, and we assume there are $t > k\sqrt{q}$ such x. But notice that $Q'(x)$ has degree at most $\sqrt{q} + \sqrt{q}(k-1) = k\sqrt{q}$, and so the only way it can have t zeros is if it is identically zero. Now, as suggested, we view Q as a univariate polynomial in y with coefficients in $\mathbb{F}_q[x]$. We have just determined that $Q(x, p(x))$ is identically zero – i.e., $p(x)$ is a root of Q. Therefore $y - p(x)$ divides Q. We can then factor Q (which as noted can be done efficiently), and be assured that every codeword $p(x)$ with agreement $t > k\sqrt{q}$ with R will appear as a factor $y - p(x)$ in this factorization.

   (c) We must show that the number of (i, j) pairs with $i + (k-1)j \leq D = \sqrt{2kq}$ is greater than q. Then using the same argument as in part (a), we will be able to find the desired $Q(x, y)$. For simplicity assume $D/(k-1)$ is an integer. Consider the set of integer pairs $R = \{(i, j) : 0 \leq i \leq D, 0 \leq j \leq D/(k-1)\}$, and observe that $|R| = D^2/(k-1) = 2kq/(k-1)$. At least half of the pairs $(i, j) \in R$ lie on or below the line $i + (k-1)j \leq D$. Thus there are at least $|R|/2 = kq/(k-1) > q$ such pairs as desired.

   Now, using the same argument as in part (b), we see that $Q'(x) = Q(x, p(x))$ has degree at most $D = \sqrt{2kq}$, and therefore if p and R have agreement $t > D$, Q' must be identically zero, and thus $p(x)$ is a root of $Q(x, y)$. The same factoring algorithm as we used above then finds all codewords p that have agreement $t > \sqrt{2kq}$ with the received word R.

4. (a) A generic polynomial $Q(x, y_0, y_1, \ldots, y_{c-1})$ with individual degrees at most $h - 1$ has the form $Q(x, y) = \sum_{m \in M} c_m m(x, y_0, y_1, \ldots, y_{c-1})$ where M is the set of monomials in $x, y_0, y_1, \ldots, y_{c-1}$ with the required degree constraints. Note that $|M| = h^{c+1} > q$. We have q constraints of the form $Q(w, R(w)) = 0$ for $w \in \mathbb{F}_q$ that we wish to satisfy. Each of these is a linear constraint on the coefficients c_m. Thus we have a homogeneous system of q linear equations in $|M| > q$ variables, so there is a non-trivial solution (i.e., one in which not all the c_m are zero). This gives us the desired polynomial Q, and because solving this system of equations can be done in a straightforward way using Gaussian elimination, Q can be found efficiently.

   (b) Consider $Q'(x) = Q(x, p^{(0)}(x), p^{(1)}(x), \ldots, p^{(c-1)}(x))$. By construction $Q'(w) = 0$ for those $w \in \mathbb{F}_q$ on which $[p^{(0)}, \ldots, p^{(c-1)}]$ and R agree, and we assume there are $t > (c+1)kh$ such w. But notice that $Q'(x)$ has degree at most $(h-1)+c(h-1)(k-1) < (c+1)kh$, and so the only way it can have t zeros is if it is the zero polynomial.

   (c) We claim that every polynomial $p : \mathbb{F}_q \to \mathbb{F}_q$ of degree at most $k - 1$ for which $[p^{(0)}, \ldots, p^{(c-1)}]$ and R agree in $t > (c+1)kh$ positions, when viewed as an element

of $\mathbb{F}_q[x]/E(x)$, is a *root* of Q^*. Why? Imagine plugging $p(x)$ into the polynomial Q^*. This is just

$$Q(x, p(x), (p(x))^h, (p(x))^{h^2}, \ldots, (p(x))^{h^{c-1}}) \bmod E(x).$$

The operation of taking a polynomial modulo $E(x)$ commutes with the operations of polynomial addition and multiplication, so this polynomial can also be written as:

$$Q(x, p(x) \bmod E(x), (p(x))^h \bmod E(x), (p(x))^{h^2} \bmod E(x), \ldots, (p(x))^{h^{c-1}} \bmod E(x)) \bmod E(x),$$

which is just $Q(x, p^{(0)}(x), p^{(1)}(x), p^{(2)}(x), \ldots, p^{(c-1)}(x)) \bmod E(x)$. In the previous part we have established that this polynomial is the zero polynomial, and so we conclude that $p(x)$ is a root of Q^* as claimed.

The number of roots of $Q^*(z)$ is bounded by its degree, which is at most $(h-1)(1 + h + h^2 + \cdots + h^{c-1}) = h^c - 1$, which is at most q. If we find all the roots of Q^*, these will include the $p(x)$ polynomials associated with every PV codeword with agreement at least $(c-1)kh$ with the received word R.