CS 151 Complexity Theory **Spring 2017**

Problem Set 6

Out: May 17 *Due:* **May 24**

Reminder: you are encouraged to work in groups of two or three; however you must turn in your own write-up and note with whom you worked. You may consult the course notes and the text (Papadimitriou). The full honor code guidelines can be found in the course syllabus.

Please attempt all problems. **To facilitate grading, please turn in each problem on a separate sheet of paper and put your name on each sheet. Do not staple the separate sheets.**

1. The following problem comes from Learning Theory, where the VC-dimension gives important information about the difficulty of learning a given concept. Given a collection $\mathcal{S} = \{S_1, S_2, \ldots, S_M\}$ of subsets of a finite set U, the *VC dimension* of $\mathcal{S}$ is the size of the largest set $X \subseteq U$ such that for every $X' \subseteq X$, there is an i for which $S_i \cap X = X'$ (we say that X is *shattered* by $\mathcal{S}$). A Boolean circuit C that computes a function $f : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}$ succinctly represents a collection $\mathcal{S}$ of 2^m subsets of $U = \{0,1\}^n$ as follows: the set S_i consists of exactly those elements x for which $C(i,x) = 1$. Finally, the language VC-DIMENSION is the set of pairs (C,k) for which C represents a collection of subsets $\mathcal{S}$ whose VC dimension is at least k.

   (a) Argue that VC-DIMENSION is in $\mathbf{\Sigma_3^p}$. Hint: what is the size of the largest possible set X shattered by a collection of 2^m subsets?

   (b) Show that VC-DIMENSION is $\mathbf{\Sigma_3^p}$-complete by reducing from QSAT_3. Hint: the universe U should be the set $\{0,1\}^\ell \times \{1,2,3,\ldots,\ell\}$. For each ℓ-bit string a, define the subset $U_a = \{a\} \times \{1,2,3,\ldots,\ell\}$. The sets in your instance of VC-DIMENSION should each be a subset of some U_a; note that the problem definition does not require that sets S_i and S_j to be different for $i \neq j$ — indeed your reduction will probably produce many copies of the same set with different "names."

2. Toda's Theorem (Part II). In this problem you will show that the ability to "count" is enough to capture the entire Polynomial-Time Hierarchy.

   (a) Let C be Boolean circuit, and let g be a polynomial with nonnegative integer coefficients. Describe a (deterministic) procedure that, given C, produces a circuit C' such that

   $$\left|\{y : C'(y) = 1\}\right| = g\left(\left|\{x : C(x) = 1\}\right|\right),$$

   and that runs in time polynomial in the size of C and the size of g (when it is encoded in the natural way as a vector of coefficients).
   Hint: given two circuits, figure out how to produce a circuit with a number of satisfying assignments equal to the *sum* or *product* of the number of satisfying assignments of the original two circuits.

(b) Give a (deterministic) procedure that, given m a power of two, outputs (as a sequence of coefficients) a polynomial $g : \mathbb{Z} \to \mathbb{Z}$ with nonnegative integer coefficients for which

$$\begin{aligned} t \equiv 0 \bmod 2 &\Rightarrow g(t) \equiv 0 \bmod 2^m \\ t \equiv 1 \bmod 2 &\Rightarrow g(t) \equiv 1 \bmod 2^m, \end{aligned}$$

and that runs in time poly(m).
Hint: first verify that the polynomial $g_0(t) = 3t^2 - 2t^3$ has the property that, for all i,

$$\begin{aligned} t \equiv 0 \bmod 2^{2^i} &\Rightarrow g_0(t) \equiv 0 \bmod 2^{2^{i+1}} \\ t \equiv 1 \bmod 2^{2^i} &\Rightarrow g_0(t) \equiv 1 \bmod 2^{2^{i+1}}. \end{aligned}$$

(c) Prove that $\mathbf{PH} \subseteq \mathbf{P}^{\#\mathbf{P}}$.
Hint: use the result proved in Problem 2(e) on the last problem set. Use Problem 2(d) to argue that any language in $\mathbf{BPP}^{\oplus\mathbf{P}}$ can be decided in $\mathbf{BPP}^{\oplus\mathbf{P}}$ *with the oracle machine making a single query and then immediately entering* q_{accept} *or* q_{reject} *depending on the answer.* Argue that the outcome of this query is determined by the number of satisfying assignments of a circuit C; now apply parts (a) and (b).

3. 2-sided versus 1-sided error for **MA** and **AM**. For this problem you may want to recall "strong error-reduction via extractors" from Lecture 11, and the proof of $\mathbf{BPP} \subseteq \mathbf{\Sigma_2^P} \cap \mathbf{\Pi_2^P}$ from Lecture 12. In the characterizations below, y, z, y', z' are all strings whose length is polynomial in $|x|$.

(a) Recall that a language L is in **MA** if there is a language R in **P** for which:

$$\begin{aligned} x \in L &\Rightarrow \exists y \text{ for which } \Pr_z[(x, y, z) \in R] \geq 2/3, \text{ and} \\ x \notin L &\Rightarrow \forall y \ \Pr_z[(x, y, z) \in R] \leq 1/3. \end{aligned}$$

Prove that for every such language L, there is a language R' in **P** for which:

$$\begin{aligned} x \in L &\Rightarrow \exists y' \text{ for which } \Pr_{z'}[(x, y', z') \in R'] = 1, \text{ and} \\ x \notin L &\Rightarrow \forall y' \ \Pr_{z'}[(x, y', z') \in R'] \leq 1/3. \end{aligned}$$

(b) Recall that a language L is in **AM** if there is a language R in **P** for which:

$$\begin{aligned} x \in L &\Rightarrow \Pr_y[\exists z \text{ for which } (x, y, z) \in R] \geq 2/3, \text{ and} \\ x \notin L &\Rightarrow \Pr_y[\exists z \ (x, y, z) \in R] \leq 1/3. \end{aligned}$$

Prove that for every such language L, there is a language R' in **P** for which:

$$\begin{aligned} x \in L &\Rightarrow \Pr_{y'}[\exists z' \text{ for which } (x, y', z') \in R'] = 1, \text{ and} \\ x \notin L &\Rightarrow \Pr_{y'}[\exists z' \ (x, y', z') \in R'] \leq 1/3. \end{aligned}$$

Hint: for both parts, use strong error reduction, and then allow Merlin to pick half of Arthur's random string.

4. The *permanent* of an $n \times n$ matrix A is defined by

$$\text{PERM}(A) = \sum_{\pi} \prod_{i=1}^{n} A[i, \pi(i)]$$

where π ranges over all permutations of the set $\{1, 2, \ldots, n\}$. This looks similar to the *determinant* which is defined by

$$\text{DET}(A) = \sum_{\pi} \prod_{i=1}^{n} \text{sgn}(\pi) A[i, \pi(i)]$$

where π ranges over all permutations of the set $\{1, 2, \ldots, n\}$, and $\text{sgn}(\pi)$ is 1 if π can be written as the product of an even number of transpositions, and -1 otherwise. The determinant of an $n \times n$ matrix can be computed in polynomial time by Gaussian elimination. In this problem you will show that the situation is (probably) dramatically different for the problem of computing the permanent.

   (a) Prove that the function f mapping square matrices A with nonnegative integer entries to $\text{PERM}(A)$ is in $\#\mathbf{P}$.

   (b) Prove that the function f is $\#\mathbf{P}$-complete. Hint: your reduction will produce matrices with all entries either 0 or 1.