CS 151 Complexity Theory **Spring 2021**

Problem Set 5

Out: May 6 *Due:* **May 13**

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 optional text (Papadimitriou). The full honor code guidelines and collaboration policy can be found in the course syllabus.

Please attempt all problems. **Please turn in your solutions via Gradescope, by 1pm Los Angeles time on the due date.**

1. CNFs and DNFs. Recall that a Boolean formula is said to be in *3-CNF* form if it is the conjunction of *clauses*, with each clause being the disjunction of at most 3 literals. A Boolean formula is said to be in *3-DNF* form if it is the disjunction of *terms*, with each term being the conjunction of at most 3 literals.

   Describe a polynomial-time computable function that is given as input a fan-in two $(\wedge, \vee, \neg)$-circuit $C(x_1, x_2, \ldots, x_n)$, and produces a 3-CNF Boolean formula ϕ on variables $x_1, x_2, \ldots, x_n$ and additional variables $z_1, z_2, \ldots, z_m$ for which for every setting of the x variables

   $$\exists z_1, z_2, \ldots, z_m \;\; \phi(x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_m) = 1 \Leftrightarrow C(x_1, x_2, \ldots, x_n) = 1.$$

   Also, describe a polynomial-time computable function that is given as input a fan-in two $(\wedge, \vee, \neg)$-circuit $C(x_1, x_2, \ldots, x_n)$, and produces a 3-DNF Boolean formula ϕ on variables $x_1, x_2, \ldots, x_n$ and additional variables $z_1, z_2, \ldots, z_m$ for which for every setting of the x variables

   $$\forall z_1, z_2, \ldots, z_m \;\; \phi(x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_m) = 1 \Leftrightarrow C(x_1, x_2, \ldots, x_n) = 1.$$

   Hint: identify the z variables with the gates of C.

2. Toda's Theorem (Part I). This problem concerns the class $\oplus\mathbf{P}$, which you will show to be quite powerful. A uniform way to define the classes $\mathbf{NP}, \mathbf{BPP}$ and the new class $\oplus\mathbf{P}$ is in terms of polynomial-time nondeterministic Turing Machines that are standardized so that they make the same number of binary nondeterministic choices on each computation path. Specifically, a language L is in:

   - **NP** iff there is such a polynomial-time nondeterministic Turing Machine M for which $x \in L$ implies that at least one of the computation paths accepts, and $x \notin L$ implies that no computation paths accept.
   - **BPP** iff there is such a polynomial-time nondeterministic Turing Machine M for which $x \in L$ implies that at least 2/3 of the computation paths accept, and $x \notin L$ implies that at most 1/3 of the computation paths accept.

- $\oplus\mathbf{P}$ iff there is such a polynomial-time nondeterministic Turing Machine M for which $x \in L$ implies that an odd number of the computation paths accept, and $x \notin L$ implies that an even number of the computation paths accept.

Below we will also be discussing the classes $\mathbf{NP}^A, \mathbf{BPP}^A$ and $(\oplus\mathbf{P})^A$. These are the classes obtained by replacing the polynomial-time nondeterministic Turing Machine M in the definitions above with a polynomial-time nondeterministic *oracle* Turing Machine M that is equipped with language A as its oracle. As usual, if we write $\mathcal{C}$ instead of A in the exponent, for some complexity class $\mathcal{C}$, we mean that *any* language $A \in \mathcal{C}$ is permitted as the oracle. If $\mathcal{C}$ has a complete language (as $\mathbf{NP}^A$ and $(\oplus\mathbf{P})^A$ do, for any oracle A), then by using that language as the oracle we can solve any instance of a problem in $\mathcal{C}$ with a single call to this specific oracle.

(a) The following is a more general restatement of the Valiant-Vazirani Theorem from Lecture 8.

Lemma 5.1 *There is a randomized procedure that receives as input an integer n, runs in poly(n) time, and outputs a poly(n)-size circuit C with the following property: for each subset $T \subseteq \{0,1\}^n$, if $|T| > 0$, then with probability at least $1/(8n)$ over the randomness of the procedure,*

$$|\{x : x \in T \text{ and } C(x) = 1\}| = 1.$$

Using this lemma, prove that for every oracle A, $\mathrm{NP}^A \subseteq \mathrm{BPP}^{(\oplus\mathbf{P}^A)}$. It may be helpful to think about the non-relativized statement first.

(b) Prove that for every oracle A,

$$\mathbf{NP}^A \subseteq \mathbf{BPP}^A \;\Rightarrow\; \mathbf{NP}^{(\mathbf{NP}^{(\mathbf{NP}^{(\ldots(\mathbf{NP}^A)\ldots)})})} \subseteq \mathbf{BPP}^A$$

in which the tower of $\mathbf{NP}$ classes has height i, for $i = 1, 2, 3, \ldots$.

Hint: first, figure out how to use error-reduction to argue that $\mathbf{NP}^{\mathbf{BPP}} \subseteq \mathbf{BPP}^{\mathbf{NP}}$ and $\mathbf{BPP}^{\mathbf{BPP}} \subseteq \mathbf{BPP}$. If you are stuck, you can take the relativized versions of these statements as given, for partial credit.

(c) Prove that co-$(\oplus\mathbf{P}) \subseteq \oplus\mathbf{P}$.

(d) Prove that $(\oplus\mathbf{P})^{\oplus\mathbf{P}} \subseteq \oplus\mathbf{P}$.

Hint: Use the fact that odd $\times$ odd = odd. For each language L in $(\oplus\mathbf{P})^{\oplus\mathbf{P}}$ with associated nondeterministic oracle TM M, you will be designing a nondeterministic TM M'. A helpful strategy is for M' to begin by nondeterministically guessing a *transcript* of M on input x, which contains a sequence of nondeterministic choices made by M, together with a sequence of queries made to the oracle, and a sequence of yes/no answers. Many of these transcripts will be inaccurate in the sense that they don't agree with the functioning of M on the specified computation path, with *the actual oracle* in $\oplus\mathbf{P}$ answering queries. Nevertheless, some nondeterministic guesses produce "correct" transcripts...

(e) Prove that $\mathbf{PH} \subseteq \mathbf{BPP}^{\oplus\mathbf{P}}$.

3. Approximate counting and sampling with an **NP** oracle. For every n, k (positive integers, with $k \leq n$), there is a multiset $\mathcal{H}_{n,k}$ of functions $h : \{0,1\}^n \to \{0,1\}^k$, called an "$n$-wise independent hash family". This multiset comes equipped with a probabilistic procedure that runs in time poly(n) and outputs a uniformly chosen h from $\mathcal{H}_{n,k}$, in the form of a circuit for h of size poly(n). These functions behave like random functions from n bits to k bits in the following sense:

   Lemma 5.2 *For every set $S \subseteq \{0,1\}^n$ and every $y \in \{0,1\}^k$:*

$$\Pr_{h \in \mathcal{H}_{n,k}} \left[|\{x : x \in S \wedge h(x) = y\}| > 2 \cdot \frac{|S|}{2^k} \right] \leq 2^{-2n},$$

   provided that $2^k \leq 4|S|/n^4$.

   Note that for a random function h from n bits to k bits, the expected size of

$$\{x : x \in S \wedge h(x) = y\}$$

   is $|S|/2^k$; the lemma says that with high probability, the same set with respect to a function h drawn uniformly from $\mathcal{H}_{n,k}$ does not exceed this expected size by more than a factor of two.

   In the problems below, the input is a set $S \subset \{0,1\}^n$ given *implicitly* by a circuit $C : \{0,1\}^n \to \{0,1\}$ for which $C(x) = 1$ iff $x \in S$. You can think of C as an instance of CIRCUIT SAT, and then the questions below concern the problems of estimating the number of satisfying assignment, and sampling from them, respectively.

   (a) Describe a probabilistic polynomial-time procedure, with access to an **NP** oracle, that with probability at least 7/8 outputs an integer k for which $2^k < \frac{|S|}{n^4} \leq 2^{k+2}$. Hint: argue that deciding whether an implicitly given set has size *at least* s, for polynomially-large s, is in **NP**, and then perform an experiment for each $k = 1, 2, 3, \ldots$.

   (b) Describe a probabilistic polynomial-time procedure, with access to an **NP** oracle, that outputs "fail" with probability at most 7/8 and otherwise outputs an exactly uniformly distributed element of S. Hint: suppose a notebook has L lines on every page, with an enumeration of the elements of a set S are written on a subset of the lines in the notebook. Consider selecting a random page and a random line on that page, and outputting the element written on that line, or "fail" if the line is empty. What is the probability of outputting a given element of S? What is the probability of outputting "fail"?