15-855: An Introduction to Computational Complexity Theory Fall 2015
Problem Set #3 V. Guruswami & M. Wootters
Due on **October 22, 2015** (in class, or by email to yuzhao1@cs.cmu.edu).

Instructions: Same as for problem set 1.

Pick any 6 problems to solve out of the 8 problems. *If you turn in solutions to more than 6 problems, we will take your top 6 scoring problems.*

1. **"#P via polynomials."** In this problem, you will give an alternate characterization of the complexity class #P. Define a "patiently multiplying arithmetic program" (PMAP) to be a program with a sequence $(p_1, p_2, \ldots, p_t)$ of instructions such that each p_k, $1 \leq k \leq t$, is of one of the following forms:
   (i) p_k is a constant 0 or 1,
   (ii) $p_k = x_i$ or $p_k = 1 - x_i$ for some $i \leq k$,
   (iii) $p_k = p_i + p_j$ for some $1 \leq i, j < k$,
   (iv) $p_k = p_i p_j$ for some i, j such that $i + j \leq k$ ("patient" multiplication)
   (v) $p_k = p_{j|x_i=0}$ or $p_{j|x_i=1}$ for some $1 \leq i, j < k$. (Here, $p_{j|x_i=0}$ means the polynomial obtained from p_j by substituting 0 for the variable x_i.)

   A PMAP as above defines a sequence of polynomials in the obvious way. A PMAP with instruction sequence $P = (p_1, p_2, \ldots, p_t)$ is said to compute the polynomial p_t, which we denote by $\tilde{P}$. A family of PMAPs $P_1, P_2, ...$ is a uniform family if each $\tilde{P}_n$ has at most n variables $x_1, x_2, \ldots, x_n$ and if there is a polynomial time deterministic Turing machine that on input 1^n prints the instructions of the PMAP P_n.

   (a) Prove that a function $f : \{0,1\}^* \to \mathbb{N}$ is in #P if and only if there is a uniform family of PMAPs $\{P_n\}_{n \geq 0}$ such that for every non-empty string $x \in \{0,1\}^*$, $f(x) = \tilde{P}_{|x|}(x)$.

   (b) Suppose the patient multiplication rule (iv) above is replaced by the perhaps more natural rule (similar to the one for addition) that $p_k = p_i p_j$ for some $1 \leq i, j < k$, and we also add the rule

   (vi) $p_k = 1 - p_i$ for some $i < k$.

   Now, what is the class of languages whose characteristic function is computed by a uniform family of arithmetic programs belonging to this new category? Give an informal/intuitive argument justifying your answer.

2. **"Definitions: wrong and right":**

   (a) Following the characterization of NP as problems whose solutions can be verified in P with the help of a certificate, we can imagine that perhaps NL can be characterized as the class of logspace verifiable languages defined as follows. Define $\widetilde{\mathsf{NL}}$ to be the set of languages A such that there is a log-space machine ("verifier") M and a constant c such that:

$$x \in A \quad \Leftrightarrow \quad \exists y \text{ with } |y| \leq c|x|^c \text{ s.t. } M(x, y) \text{ accepts.}$$

Show that $\widetilde{\mathsf{NL}} = \mathsf{NP}$.

(b) Suppose we now restrict the verifier M to have only left-to-right read-once access to the certificate y. In other words, the verifier is given x on the read-only input tape and the certificate y on a separate read-only tape in which the head can never move left. In addition, M has a constant number of read/write tapes each with $O(\log |x|)$ cells.

Prove that a language A has such a restricted logspace verifier if and only if $A \in \mathsf{NL}$.

(c) Here is a restatement of the definition of RL: A language A is in RL if there is a log-space, poly-time algorithm, with one-way read-only access to a tape of random bits, which accepts strings in A with probability at least $1/2$ and accepts strings not in A with probability 0. Define $\widetilde{\mathsf{RL}}$ to be the same class except that the condition of running in poly-time is dropped. Show that $\widetilde{\mathsf{RL}} = \mathsf{NL}$.

3. **"Fun with MA."** For the purposes of this problem, fix the definition of MA as follows: $L \in \mathsf{MA}$ if there is a predicate $R(x, w, r) \in \mathsf{P}$ (with $|w|, |r| = \text{poly}(|x|)$) such that:

$$x \in L \Rightarrow \exists w \, \mathop{\mathbf{Pr}}_{r}[R(x, w, r) = 1] = 1, \tag{1}$$

$$x \notin L \Rightarrow \forall w \, \mathop{\mathbf{Pr}}_{r}[R(x, w, r) = 1] \leq 1/2. \tag{2}$$

a) Explain briefly why the $1/2$ in (2) can be made $1/2^{n^c}$ for any fixed constant c.

b) Show $\mathsf{MA} \subseteq \mathsf{PP}$.

c) Suppose we replaced the $= 1$ in (1) with $\geq 2/3$. Show this does not change the definition of MA.

d) Show that $\mathsf{BPP} \subseteq \mathsf{MA}$. How does this compare with the inclusion you proved in Problem 6 on Homework 1?

4. **"Checkers."** Imagine you have black-box oracle access to a program that supposedly computes the function $f : \{0,1\}^* \to \{0,1\}^*$ (which has the property that $|f(x)| \leq \text{poly}(|x|)$). However, you are a skeptic, and are concerned that it might be computing some different function, $\tilde{f}$. Nevertheless, you would like to be able to compute f confidently.

We say that a function f has a *checker* if there is a BPP algorithm $\mathcal{A}^?$ with oracle access to some function such that

(i) For every function $\tilde{f}$ used as the oracle, $\mathcal{A}^{\tilde{f}}(x)$ always halts in polynomial time and outputs either a string or "fail".

(ii) If f is used as the oracle, then for all x, $\mathcal{A}^f(x) = f(x)$ with probability 1.

(iii) No matter what function $\tilde{f}$ is used as the oracle, and for every x, $\mathbf{Pr}[\mathcal{A}^{\tilde{f}}(x) \in \{f(x), \mathsf{fail}\}] \geq 2/3$. Here the probability is over the randomness of the BPP algorithm $\mathcal{A}$.

We say that a language L has a checker if the function $f(x)$ which is 1 if $x \in L$ and 0 otherwise has a checker.

Now to your questions:

a) Let f be the function which, on input two matrices $A, B \in \mathbb{F}_2^{n \times n}$, outputs AB. Show that f has a checker which runs in time $O(n^2)$.

b) Show that the language Graph-Non-Isomorphism has a checker.

c) Show one of the following (your choice): Permanent has a checker; or, TQBF has a checker. (Hint: use the proofs that $\mathsf{P}^{\#\mathsf{P}}, \mathsf{PSPACE} \subseteq \mathsf{IP}$.)

5. **"More rounds don't help."** For the purposes of this problem, you may assume the following definitions of AM and MA (and it builds character and/or has already appeared on this problem set to verify that they are they same as all the other definitions):

   - $L \in \mathsf{MA}$ if there is a predicate $R(x, w, r) \in \mathsf{P}$ (with $|w|, |r| = \text{poly}(|x|)$) such that:

   $$x \in L \Rightarrow \exists w \, \mathop{\mathbf{Pr}}_{r}[R(x, w, r) = 1] = 1,$$

   $$x \notin L \Rightarrow \forall w \, \mathop{\mathbf{Pr}}_{r}[R(x, w, r) = 1] \leq 1/2^{|x|^c}$$

   for any constant c.

   - $L \in \mathsf{AM}$ if there is a predicate $R(x, w, r) \in \mathsf{P}$ (with $|w|, |r| = \text{poly}(|x|)$) such that:

   $$x \in L \Rightarrow \mathop{\mathbf{Pr}}_{r}[\exists w, R(x, w, r) = 1] = 1,$$

   $$x \notin L \Rightarrow \mathop{\mathbf{Pr}}_{r}[\forall w, R(x, w, r) = 1] \leq 1/2.$$

   a) Show that $\mathsf{MA} \subseteq \mathsf{AM}$.

   b) Extend your approach from (a) to show that $\mathsf{MA}[k+1] \subseteq \mathsf{AM}[k]$ for any constant k.

   c) Conclude that $\mathsf{AM}[k] = \mathsf{AM}[2]$ for all constants k. Why does this approach not work if k is not constant?

6. **"Collapses."** Show:

   a) If $\mathsf{NP} \subseteq \mathsf{P/poly}$ then $\mathsf{AM} = \mathsf{MA}$.

   b) If a language L has a checker, and $L \in \mathsf{P/poly}$, then $L \in \mathsf{MA}$. Conclude that $\mathsf{PSPACE} \in \mathsf{P/poly} \Rightarrow \mathsf{PSPACE} = \mathsf{MA}$.

   c) If $\mathsf{coNP} \subseteq \mathsf{AM}$, then $\mathsf{PH} = \Sigma_2$. (Hint: first show that $\mathsf{AM} \subseteq \Pi_2$, and then show that $\mathsf{coNP} \subseteq \mathsf{AM} \Rightarrow \Sigma_2 \subseteq \mathsf{AM}$).

7. **"Public coins for GNI"**

   a) Let $L \in \mathsf{NP}$ and $k \in \mathbb{N}$. Arthur and Merlin are both given k as input. Give an AM protocol so that

   $$|\{L \cap \{0,1\}^n\}| \geq k \Rightarrow \mathbb{P}\{\text{Arthur accepts}\} \geq 2/3$$

   and

   $$|\{L \cap \{0,1\}^n\}| \leq k/2 \Rightarrow \mathbb{P}\{\text{Arthur rejects}\} \leq 1/3.$$

   (Hint: you may assume the existence of efficiently computable pair-wise independent hash families.)

   b) Use your solution from (a) to give an AM protocol for Graph Non-Isomorphism.

   (Hint: Consider the set $\{H \ : \ H \simeq G\}$ for a graph G, and use part (a).)

8. **"An XOR-style lemma."** Below, we denote by U_ℓ the uniform distribution on $\{0,1\}^\ell$.

   (a) Suppose $f_1 : \{0,1\}^n \to \{0,1\}$ and $f_2 : \{0,1\}^m \to \{0,1\}$ are two functions such that for all circuits C_1 (resp. C_2) of size $s_1(n)$ (resp. $s_2(m)$), we have that $\mathbf{Pr}_{x \leftarrow U_n}[C_1(x) = f(x)] \leq p_1(n)$ and $\mathbf{Pr}_{y \leftarrow U_m}[C_2(y) = f_2(y)] \leq p_2(m)$. Assume that $m \geq n$.
   Prove that for all circuits C outputting a pair of bits of size

$$S = \min\{\frac{s_1(n)}{\text{poly}(m/\epsilon)}, s_2(m) - O(m)\}$$

   it is the case that

$$\mathbf{Pr}_{(x,y) \leftarrow U_n \times U_m}[C(x,y) = (f_1(x), g_2(y))] \leq p_1(n)p_2(m) + \epsilon \ . \tag{3}$$

   (Hint: Assuming the existence of C with better accuracy than (3), find a circuit of size C' that correctly computes $f_1(x)$ with probability exceeding $p_1(n)$. To do this, hardwire the values $f_2(y_i)$ on a sample of $\text{poly}(m, \epsilon)$ points y_i.)

   (b) Suppose $f : \{0,1\}^n \to \{0,1\}$ is such that for every circuit C of size s, $\mathbf{Pr}_{x \leftarrow U_n}[C(x) = f(x)] \leq p(n)$. Define the function $f^{(k)} : (\{0,1\}^n)^k \to \{0,1\}^k$ as $f^{(k)}(x_1, x_2, \ldots, x_k) = (f(x_1), f(x_2), \ldots, f(x_k))$. Prove that for every circuit $\tilde{C}$ (with k output bits) of size $s' \leq s \cdot \text{poly}(\epsilon/n)$,

$$\mathbf{Pr}_{\overline{x}=(x_1,x_2,\ldots,x_k) \leftarrow (U_n)^k}[\tilde{C}(\overline{x}) = f^{(k)}(\overline{x})] \leq p(n)^k + \epsilon \ .$$

   (Hint: Use (8a) inductively.)