CS 221: Computational Complexity **Prof. Salil Vadhan**

Problem Set 4

Assigned: Wed. Mar. 24, 2010 Due: Thu. Apr. 8, 2010 (5 PM sharp)

- You must *type* your solutions. LaTeX, Microsoft Word, and plain ascii are all acceptable. Submit your solutions *via email* to `cs221-hw@seas.harvard.edu`. If you use LaTeX, please submit both the compiled file (`.pdf`) and the source (`.tex`). Please name your files `PS4-yourlastname.*`.
- Strive for clarity and conciseness in your solutions, emphasizing the main ideas over low-level details. Do not despair if you cannot solve all the problems! Difficult problems are included to stimulate your thinking and for your enjoyment, not to overwork you. *'ed problems are extra credit.

Problem 1. (one-sided error vs. two-sided error) Show that if $\mathbf{NP} \subseteq \mathbf{BPP}$, then $\mathbf{NP} = \mathbf{RP}$.

Problem 2. (Cook reductions to promise problems) Note that for a promise problem Π, "running an algorithm with oracle Π" is not in general well-defined, because it is not specified what the oracle should return if the input violates the promise.[1] Thus, when we say that a problem Γ can be solved in polynomial time with oracle access to Π, we mean that there is a polynomial-time oracle algorithm A such that for *every* oracle $O : \{0,1\}^* \to \{0,1\}$ that solves Π (i.e. O is correct on $\Pi_Y \cup \Pi_N$), it holds that A^O solves Γ.

1. Let Π be the promise problem

$$\begin{aligned} \Pi_Y &\stackrel{\text{def}}{=} \{(\varphi,\psi) : \varphi \in \text{SAT}, \psi \notin \text{SAT}\} \\ \Pi_N &\stackrel{\text{def}}{=} \{(\varphi,\psi) : \varphi \notin \text{SAT}, \psi \in \text{SAT}\} \end{aligned}$$

   Show that $\Pi \in \mathbf{prNP} \cap \mathbf{prcoNP}$ but $\text{SAT} \in \mathbf{prP}^{\Pi}$.

   This implies that $\mathbf{prNP} \subseteq \mathbf{prP}^{\mathbf{prNP} \cap \mathbf{prcoNP}}$. Note that an analogous inclusion seems unlikely for language classes, since $\mathbf{P}^{\mathbf{NP} \cap \mathbf{coNP}} = \mathbf{NP} \cap \mathbf{coNP}$, as shown in an earlier section.

2. (*) Show that $\mathbf{prBPP} \subseteq \mathbf{prRP}^{\mathbf{prRP}}$, and thus $\mathbf{prRP} = \mathbf{prP}$ iff $\mathbf{prBPP} = \mathbf{prP}$. (Hint: look at the proof that $\mathbf{BPP} \subseteq \mathbf{PH}$.)

[1] A similar issue comes up with problems where there are multiple valid answers on a given input, such as search or approximation problems. Again, in such cases, we should require that the algorithm works correctly for every oracle that solves the problem.

Problem 3. (#MATCHINGS and #INDEPENDENT SETS)

1. A *matching* in a graph is a set S of edges such that every vertex touches *at most one* edge in S (as opposed to exactly one, as required in a perfect matching). Show that #MATCHINGS, the problem of counting all the matchings in a graph, is #P-complete. (Hint: reduce from #PERFECT MATCHINGS. Given a graph G, consider the graph G_k obtained by attaching k new edges to each vertex of G. G_k has $n + nk$ vertices, where n is the number of vertices in G. Show that the number of perfect matchings in G can be recovered from the number of matchings in each of $G_0, \ldots, G_n$.)

2. An *independent set* in a graph G is a set S of vertices such that no two elements of S are connected by an edge in G. Prove that #INDEPENDENT SETS, the problem of counting the number of independent sets in a graph, is #**P**-complete.

3. Prove that a fully polynomial randomized approximation scheme for #MATCHINGS implies a fully polynomial almost-uniform sampler for MATCHINGS. (This is the converse of what we showed in class.)

4. Show that approximating #INDEPENDENT SETS to within any constant factor is **NP**-hard. In contrast, there are a fully polynomial randomized approximation schemes known for #PERFECT MATCHINGS and #MATCHINGS.

Problem 4. (parallel search vs. decision)

1. Recall that we can solve the SAT-SEARCH problem in polynomial time given an oracle for deciding SAT. Note that this reduction algorithm makes *adaptive* queries to the SAT oracle, i.e. its i'th query depends on the answers to its first $i - 1$ queries. Show that the reduction can be made nonadaptive if we allow it to be randomized. (Hint: Use Valiant-Vazirani)

2. Suppose that there is a $1/\mathrm{poly}(n)$-hard **NP** search problem S (see PS0 for definition). That is, there is a polynomial p such that for every probabilistic polynomial-time machine A and sufficiently large n,
$$\Pr_{x \leftarrow \{0,1\}^n}[A(x) \notin S(x)] \geq 1/p(n).$$
Show that there is a $1/\mathrm{poly}(n)$-hard language in **NP**.