CS 301 Languages and Automata, UIC
Fall 2012, Assignment 7
Due: Friday, November 30, 2012 at start of discussion session

Unless otherwise noted, the alphabet for all questions below is assumed to be $\Sigma = \{0, 1\}$. This assignment focuses on the complexity classes P and NP, as well as polynomial-time reductions.

1. [4 marks] This question tests an important subtlety in the definition of "polynomial-time". One of the most famous open problems in classical complexity theory is whether the problem of factoring a given integer N into its prime factors is solvable in polynomial time[1]. Given positive integer N as input, why is the following naive approach to the factoring problem not polynomial-time?

   Set $m := 2$.
   Set $S := \emptyset$ for S a multi-set.
   while $m \leq \lceil \sqrt{N} \rceil$ **do**
       if m divides N **then**
           Set $S \cup \{m\}$.
           Set $N = N/m$.
       else
           Set $m = m + 1$.
       end if
   end while
   Return the set S of divisors found.

2. This question investigates the closure properties of P and NP.

   (a) [6 marks] Show that P is closed under union, concatenation, and complement.

   (b) [6 marks] Show that NP is closed under union and concatenation. Why is it not obvious that NP is closed under complement?

   (c) [4 marks] The complement of NP is called co-NP. In other words, intuitively, co-NP is the class of languages L for which if input $x \notin L$, then there is an efficiently verifiable proof of this fact. Saying NP is closed under complement is hence formally expressed as NP $=$ co-NP.

   Unfortunately, determining whether NP $=$ co-NP is arguably an even bigger open question than whether P $=$ NP. Specifically, show that if NP $\neq$ co-NP, then P $\neq$ NP.

3. [6 marks] In class, we have focused on *decision* problems, i.e. deciding whether $x \in L$ or not. For example, given a 3-CNF formula ϕ, recall that the decision problem 3SAT asks whether ϕ is satisfiable or not. In practice, however, we may not just want a YES or NO answer, but also an actual assignment which satisfies ϕ whenever ϕ is satisfiable. It turns out that in certain settings, being able to solve the *decision* version of the problem (e.g. 3SAT) in polynomial time implies we can also solve the *search* version (e.g. find the satisfying assignment itself) in polynomial time. Problems satisfying this property are called *self-reducible*.

[1] In fact, many popular cryptosystems are based on the assumption that this problem is *not* efficiently solvable. Recall from class, however, that in 1994 it was shown by Peter Shor that *quantum* computers can efficiently solve this problem!

Your task is as follows: Show that 3SAT is self-reducible. In other words, given any 3-CNF formula ϕ and a polynomial-time black-box M for determining if ϕ is satisfiable, show how to find a satisfying assignment for ϕ in polynomial time. You may assume that M is able to decide all CNF formulae with *at most* 3 variables per clause. (In class, we always assumed 3-CNF formula have precisely 3 variables per clause, but this is actually a special case of the more generally used definition of a 3-CNF formula.)

4. The remaining questions in the assignment get to the heart of Chapter 7: Polynomial-time reductions. In class, we introduced the language

$$\text{CLIQUE} = \{\langle G, k\rangle \mid G \text{ is a graph containing a clique of size at least } k\}.$$

Consider now two further languages:

$$\begin{aligned} \text{INDEPENDENT-SET} &= \{\langle G, k\rangle \mid G \text{ is a graph containing an independent set of size at least } k\} \\ \text{VERTEX-COVER} &= \{\langle G, k\rangle \mid G \text{ is a graph containing a vertex cover of size at most } k\}. \end{aligned}$$

Here, for graph $G = (V, E)$, an *independent set* $S \subseteq V$ satisfies the property that for any pair of vertices $u, v \in S$, $(u, v) \notin E$. A *vertex cover* $S \subseteq V$ satisfies the property that for any edge $(u, v) \in E$, at least one of u or v must be in S.

   (a) [4 marks] Prove that CLIQUE $\leq_p$ INDEPENDENT-SET.

   (b) [6 marks] Prove that INDEPENDENT-SET $\leq_p$ VERTEX-COVER.

5. [8 marks] Let DOUBLE-SAT $= \{\langle \phi\rangle \mid \phi \text{ has at least two satisfying assignments}\}$. Show that DOUBLE-SAT is NP-complete. Note that ϕ here is an arbitrary Boolean formula, i.e. not necessarily in CNF form.

6. Let $CNF_k = \{\langle \phi\rangle \mid \phi \text{ is a satisfiable CNF-formula where each variable appears in at most } k \text{ places}\}$.

   (a) [Bonus, 6 marks] Show that $CNF_2 \in P$.

   (b) [8 marks] Show that CNF_3 is NP-complete.

7. [8 marks] A subset of the nodes of a graph G is a *dominating set* if every other node of G is adjacent to some node in the subset. Let

$$\text{DOMINATING-SET} = \{\langle G, k\rangle \mid G \text{ has a dominating set with } k \text{ nodes}\}.$$

Show that DOMINATING-SET is NP-complete by giving a reduction from VERTEX-COVER. You may use the following slightly different (but still NP-complete) version of VERTEX-COVER:

$$\text{VERTEX-COVER} = \{\langle G, k\rangle \mid G \text{ has a vertex cover with } k \text{ nodes}\}.$$

8. [Bonus, 8 marks] Recall from class we mentioned that it's possible for a problem to be NP-hard without actually being in NP itself. Show that the language $HALT = \{\langle M, x\rangle \mid M \text{ is a TM which halts on input } x\}$ is NP-hard.