CMSC 303 Introduction to Theory of Computation, VCU
Spring 2017, Assignment 7
Due: Thursday, April 27, 2017 in class

Total marks: 52 marks + 5 marks bonus for typing your solutions in LaTeX using the provided A7 template file.

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. [12 marks]

   (a) [4 marks] Let $f(n) = 4n^2 - 30n + 6$. Prove that $f(n) \in O(n^2)$. In your proof, give explicit values for c and n_0.

   (b) [4 marks] Let $f(n) = n^{999}$ and $g(n) = (\sqrt{\log n})^{\sqrt{\log n}}$. Precisely one of the following two claims is true: $f(n) \in O(g(n))$, $g(n) \in O(f(n))$. Prove whichever one is true. In your proof, give explicit values for c and n_0. (Hint: Note that an arbitrary function $f(n)$ can be rewritten as $2^{\log_2(f(n))}$.)

   (c) [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 on a classical computer[1]. Given positive integer N as input, why is the following naive approach to the factoring problem not polynomial-time?

```
1:  Set $m := 2$.
2:  Set $S := \emptyset$ for $S$ a multi-set.
3:  **while** $m \leq N$ **do**
4:      **if** $m$ divides $N$ **then**
5:          Set $S \cup \{m\}$.
6:          Set $N = N/m$.
7:      **else**
8:          Set $m = m + 1$.
9:      **end if**
10: **end while**
11: Return the set $S$ of divisors found.
```

2. [6 marks] Let co-NP denote the complement of 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, and if $x \in L$, then no proof can cause the verifier to accept.

   Prove that if P $=$ NP, then NP is closed under complement, i.e. NP $=$ co-NP. How can closure of NP under complement hence potentially be used to resolve the P vs NP question?

3. [10 marks] In class, we introduced the language

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

[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!

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. (Hint: Given a graph G, think about its *complement*. Google "complement graph" for a definition.)

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

4. [8 marks] Let $CNF_k = \{\langle\phi\rangle \mid \phi$ is a satisfiable CNF-formula where each variable appears in at most k places$\}$. Show that CNF_3 is NP-complete. (Hint: Suppose a variable x appears (say) twice. Replace the first and second occurrences of x with new distinct variables y_1 and y_2, respectively. Next, add clauses to ensure that $y_1 = y_2$. To do this in CNF form, recall that $y_1 = y_2$ is logically equivalent to $(y_1 \implies y_2) \wedge (y_2 \implies y_1)$, and that $(a \implies b)$ can be rewritten as $(a \vee \bar{b}) \wedge (\bar{b} \vee a)$. How can this trick be applied more generally when x appears $m > 2$ times?)

5. [8 marks] It is possible for a problem to be NP-hard without actually being in NP itself. For example, the halting problem from class is clearly not in NP, since it is undecidable. However, in this question, you will show that the language $HALT = \{\langle M, x\rangle \mid M$ is a TM which halts on input $x\}$ is NP-hard. Is the halting problem hence NP-complete?

6. [8 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*.

   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.