NP-COMPLETENESS STUDY QUESTIONS
Fundamental Algorithms, Fall 2004, Professor Yap

Due: Do not submit.

INSTRUCTIONS:

- These Study Questions are in lieu of Homework 6, and focuses on NP-Completeness (Lecture XXX). This should be your priority in studying Lecture XXX. We want you to be familiar with the concept of reducibility, and to be able to do simple reductions. In particular, reducing problems to the satisfiability problem SAT.

---

1. Prove Lemma 1 (p.7), which shows that the set of well-formed Boolean formulas is in P .

   HINT: You must describe a deterministic Turing machine that accepts iff the input has the form $\#F$ for some Boolean formula F. Moreover, M must run in polynomial time. This is an exercise in programming Turing machines!

2. Exercise 4.2 (p. 10), showing that $L \in NP$ iff L is verified in polynomial time by a verification machine.

   HINT 1: As usual, an "iff" results require two demonstrations, the "if" part and the "only if" part. In the Turing machine transition

   $$(u \overset{(a,a',D)}{\longrightarrow} v) \tag{1}$$

   we call (u, a) the precondition of the transition. Note that the Turing machine must satisfy this precondition (i.e., be in state u and scanning symbol a) in order to be able to execute this transition. A nondeterministic Turing machine M is said to be in **nondeterministic normal form** if it has this property: *for each precodition* (u, a) *there are exactly two transitions with this precondition.* It is not hard to convert any nonterministic Turing machine to this normal form; furthermore, the running time of the normalized Turing machine on any input is $O(T(n))$ if the original running time is $T(n)$.

   By some convention, we can specify these two transitions as the "first" and the "second" transition for (u, a). Hence, a computation of M on any input is completely deterministic once we also know the sequence of choices (i.e., either the first or second transition is taken) made in every step!

   HINT 2: A simple Turing machine M can simulate a vertification machine V as follows. Note that V has two tapes (and two independent tape heads). We let M's tape be organized into 3 "tracks". Two of the tracks correspond to the 2 tapes of V, and the third track is for book keeping. Since V has 2 tape heads and M has only one head, in general, M must take $\Theta(T(n))$ steps for each step of V. Hence the simulation will take $O(T(n)^2)$ time.

3. The proof of Lemma 5 lists three propositions (1)-(3) that must be constructed. We only described (1). Please do the same for (2) and (3).

   HINT: the formula for (3) must depend on the graph G.

4. Let the addition predicate $A(a, b, c)$ where a, b, c represent numbers in binary notation, and $A(a, b, c)$ is true iff $a + b = c$. Show how to construct in polynomial-time a Boolean formula $F(a, b, c)$ that is true iff $A(a, b, c)$ is true.

   HINT: Assume a and b are m-bit binary numbers. So let $a = (a_1, \ldots, a_m)_2$, $b = (b_1, \ldots, b_m)_2$ and $c = (c_0, c_1, \ldots, c_m)_2$. where a_m, b_m, c_m are the least significant bit of the respective binary notations. The formula $F(a, b, c)$ involves these a_i's, b_j's and c_k's. We can introduce the sequence of carries $d = (d_0, d_1, \ldots, d_m)$ where $d_m = 0$ and $d_0 = c_0$ and d_i's is the carry bit into column i. Thus we have $a + b = c$ iff for all $i = 1, \ldots, m$,

   $$a_i + b_i = c_i + 2d_{i-1} + d_i.$$

   This equation can be expressed as a boolean formula. E.g., $a = (1, 0, 0, 1, 1)$, $b = (1, 1, 0, 0, 1)$, $c = (1, 0, 1, 1, 0, 0)$ and $d = (1, 0, 0, 1, 1, 0)$. This gives the formula

   $$G(a, b, c, d) = \bigwedge_{i=1}^{m} \text{“}(a_i + b_i = c_i + 2d_{i-1} + d_i)''$$

© Chee-Keng Yap

Since the d_i's are not given, you need to replace them by 0's and 1's, and take a disjunction over different copies of $G(a, b, c, d)$:

$$F(a, b, c) = \bigvee_d G(a, b, c, d).$$

Unfortunately, this leads to an exponential size formula as there are 2^m possible choices for d. To get around this, use a recursive formulation where we may assume m is a power of 2.