CS 351: Problem Set 2
(assigned 10/10/2002; due 10/17/2002)

On each problem, you may work with at most one other person, but you must write up the solutions independently. Please turn in each problem on a separate sheet of paper that also contains your name, the name of your collaborator (if any), and other sources (eg., papers, books) that you consulted.

(1) Consider the 3-COLORABILITY problem that is known to be NP-complete. The proof checking problem associated with 3-COLORABILITY is the following: Given a graph $G = (V, E)$ and a mapping $\chi : V \longrightarrow \{r, g, b\}$, verify that χ is a legal coloring of the vertices of G, that is, for all $(u, v) \in E$, $\chi(u) \neq \chi(v)$. When G is presented on a read-only, two-way input tape (say, as a list of edges) and χ is presented on a read-only, one-way proof tape (say, as a sequence of $|V|$ colors, one per vertex), it is possible to show that there is no algorithm using logarithmic workspace that correctly performs the verification task. Therefore, NP $\neq$ NL.

Where is the flaw in this reasoning? (For extra credit, show the impossibility of the aforementioned proof verification task.)

(2) *Equivalence between search and decision.* Suppose we have a polynomial-time algorithm A that solves the Boolean satisfiability problem, that is, the decision problem SAT. Construct a polynomial-time algorithm (using A as a sub-routine) that takes a Boolean formula φ as input and produces a satisfying assignment for φ, if one exists, or reports that none exists. How many calls to A does your algorithm make?

(*Hint.* Let $x_1, \ldots, x_n$ denote the variables of φ. Run A on φ. If $A(\varphi) =$ "no," report that no satisfying assignment exists; if $A(\varphi) =$ "yes," then run A on two more formulas $\varphi[0]$ and $\varphi[1]$, where $\varphi[b]$ is the formula obtained from φ by setting $x_1 = b$. If φ is satisfiable, then one of these two formulas must be satisfiable, and A will tell us which one. Figure out the natural way to extend this idea.)

(3) Let R be a binary relation, and let S_R denote the search problem for R and let E_R denote the corresponding existence problem. Assuming that E_R is NP-complete, show that there is a polynomial-time reduction from S_R to E_R.

(*Hint.* The idea is to mimic the satisfiability example. First define a relation R' and give an unconditional polynomial-time reduction from S_R to $E_{R'}$. Then use the fact that if E_R is NP-hard, then there is a reduction from $E_{R'}$ to E_R; note that to do this, you must place $E_{R'}$ in NP, for which you will need that E_R is in NP. Thus you will use both parts of the hypothesis that E_R is NP-complete.)

The message of this problem is that for all NP-complete problems, the search problem is reducible in polynomial time to the corresponding decision (existence) problem.