Homework 8
Due Mar 7, 2012

Prof. Amit Chakrabarti
Computer Science Department
Dartmouth College

Note: This homework is optional and will not be graded. But do discuss it and work out the problems for your own edification and for preparation for the final exam.

You may use, without proof, any result from Chapter 7 of Sipser's book.

1. Describe the error in the following fallacious "proof" that P $\neq$ NP. Consider an algorithm for SAT: "On input ϕ, try all possible assignments to the variables. Accept if any satisfy ϕ." This algorithm clearly requires exponential time. Thus, SAT has exponential time complexity. Therefore SAT is not in P. Because SAT is in NP, it must be the case that P $\neq$ NP.

2. Let DOUBLESAT $= \{\langle\phi\rangle : \phi \text{ has at least two satisfying assignments}\}$. Show that DOUBLESAT is NP-complete.

3. A burglar enters a room full of goodies with his knapsack and has to figure out what items to pack into it before running off with his loot. He wants to maximize the total worth of his loot, of course, but is constrained by the fact that his knapsack can only hold objects with total weight upto a certain maximum. If he overloads the knapsack, it will be ripped apart and all his loot will be lost. The burglar's little problem can be abstracted into a computational problem of great importance (not just to burglars): the KNAPSACK problem.

   Formally, an input to KNAPSACK consists of a number n of objects, integer weights $w_1, \ldots, w_n$ of the objects, integer values $v_1, \ldots, v_n$, and a maximum weight capacity W. The goal is to pick a subset of objects with total weight $\leq W$ while maximizing the total value of this subset. To turn this into a decision (yes/no) problem, we specify a target value V and ask if it is possible to achieve or exceed this target. Thus:

$$\text{KNAPSACK} = \left\{ \langle n, w_1, \ldots, w_n, v_1, \ldots, v_n, W, V\rangle : \exists\, S \subseteq \{1, \ldots, n\} \left(\sum_{i \in S} w_i \leq W \bigwedge \sum_{i \in S} v_i \geq V \right) \right\}.$$

   3.1. Intuitively, the burglar should prefer objects that have high value *per unit weight*. Thus, it is tempting to try to solve KNAPSACK using the following algorithm: (1) sort the objects in decreasing order of v_i/w_i and (2) consider the objects in this sorted order and, for each object considered, add it to the knapsack if the total weight after the addition would stay $\leq W$. Prove that this algorithm, although polynomial time, fails to solve KNAPSACK. (Provide a *specific* counterexample.)

   3.2. Prove that KNAPSACK is NP-complete. No wonder that algorithm did not work!

4. Prove that MAXCUT is NP-complete by solving problems 7.24 and 7.25 from Sipser's book.

5. Prove that if P were equal to NP, then the RSA cryptosystem could be broken in polynomial time. (First state a precise mathematical problem that could be solved in polynomial time and explain why solving it would lead to breaking the cryptosystem. Then prove that your mathematical problem *can* be solved in polynomial time, assuming P $=$ NP.)