CSE 200
Computability and Complexity
Homework 3
NP, Completeness, and Reductions
Part 2
Due Friday, May 26

May 2, 2006

Give proofs for each problem. Proofs can be high-level, but be precise. You may use without giving a proof any result proved in class or in the textbook. In particular, to prove NP-completeness, it suffices to give a reduction from any of the NP-complete problems from the text or from class. However, you must show your reduction is valid, by showing the equivalence of the constructed instance and the original.

1. Problem 7.30 from the Sipser text, page 298.
2. Prove that the following problem is NP-hard: Quardratic programming: Given a set of equations in n variables $x_1, ..x_n$ of the form: $\sum_{i=1}^{i=n} a_i {x_i}^2 + \sum_{i=1}^{i=n} b_i x_i = c$ where a_i, b_i, c are given integers. Decide if there is a solution, i.e., an assinment of real numbers $v_1, ..v_n$ to the variables making all equations true. Is this problem in NP? (Don't give a formal proof for whether the problem is in NP) (Hint: consider the equation: ${x_i}^2 = x_i$.)

Sudoku We started looking at reducing sudoku problems to SAT last assignment.

The *sudoku* problem of size n is as follows. The input is an $n^2 \times n^2$ matrix M whose entries are either "blank" or an integer between 1 and n^2. A solution fills in the blank spaces with integers between 1 and n^2. The following constraints must be met: Each integer from 1 to n^2 appears exactly once in each row, in each column, and in each $n \times n$ sub-matrix of the form $M[jn + 1...(j + 1)n][in + 1..(i + 1)n]$ for each $0 \leq i, j \leq n - 1$. The problem is to find any solution meeting the constraints, or return "no solution possible" if there is no such solution.

Last assignment, you gave at least two different ways to reduce the Sudoku problem to $CNF-SAT$.

Investigate at least two publically available SAT solvers, e.g., CHAFF. For each way of reducing to SAT, how hard are the resulting SAT formulas for these solvers? Use two test distributions: First, use Sudoku problems from the Marketplace section of the San Diego Union Tribune of various difficulty ratings. (Include at least one Sunday puzzle with $n = 4$). Second, test the results on randomly generated puzzles as follows:

Pick k random distinct array entries, and assign them random values from 1 to n^2.

For each method and solver, and as many values of n and k, as you can get to run in a reasonable amount of time, and averaging over as many runs as is reasonable, give the probability of a solution for n and k, and the average time your algorithm took to run. What seem to be the hardest random instances of sodoku?

(Note: Be careful not to use up too much computer time. Don't leave programs running unsupervised too long. Depending on your algorithm, you may find even very small sizes take huge amounts of time. However, some credit will be based on getting results for larger n.)