

CSE200: Computability and Complexity

Fall 2008, Final Exam

Instructor: Daniele Micciancio
Due Mon. December 8, 2008

This final exam is an individual assignment. Each student should write his/her own solutions independently. You can discuss the problems with other students to some extent (e.g., the other students you were working with on the homework assignments), but any such discussion should be clearly disclosed at the beginning of your solutions. E.g., your solutions should start with a brief paragraph saying “I discussed problem 2 with students A and B. Discussion was only aimed at understanding the questions posed in the final”, or “The method to solve problem 1.b was suggested by student A”, or “I reminded the other students about the padding method, and explained how we used it in class.” etc.

Problem 1: NP-completeness, NP, coNP

Consider the following variant of the satisfiability problem: You are given a boolean formula ϕ in CNF form that contains only positive literals (that is, each literal equals x_i for some variable, rather than \bar{x}_i). We will call this form Positive Conjunctive Normal Form (PCNF for brevity) and the corresponding formulas PCNF formulas. For PCNF formulas the satisfiability question is trivial (we can simply set all variables to TRUE and the formula will always be satisfied). Here we are interested in finding satisfying assignments that assign to true the least possible number of variables. The computational problem of finding such “minimal” satisfying assignment is modeled by the following language:

$$PSAT = \{(\phi, k) \mid \phi \text{ is a PCNF formula and } \exists \text{ satisfying assignment that sets at most } k \text{ variables to true}\}$$

(a) Prove that PSAT is NP-Complete. (Remember, this requires showing both that PSAT is in NP, and that any other NP problem reduces to PSAT in polynomial time.)

(b) Let ϕ be a PCNF formula. We define $M(\phi)$ as the minimum number of variables that need to be set to true in order for ϕ to be satisfiable. More formally $M(\phi)$ is such that $(\phi, M(\phi)) \in PSAT$ and $(\phi, k) \notin PSAT$ for all $k < M(\phi)$. Consider now the Smaller Minimum Positive Satisfying Assignment Problem (SMPSA for brevity)

$$SMPSA = \{(\phi_1, \phi_2) \mid \phi_1, \phi_2 \text{ are PCNF formulas and } M(\phi_1) < M(\phi_2)\}$$

Show that $SMPSA \in NP$ if and only if $NP = coNP$.

*[Hint: there are many possible ways to structure a solution to this problem. It is probably a good idea to break it into two parts: show that if $NP=coNP$ then $SMPSA \in NP$, and then show that if $SMPSA \in NP$ then $NP=coNP$. Another thing that you may find useful is showing that SMPSA is equivalent (e.g., via polynomial time Turing reductions) to some other problem, e.g., what happens if you change the “<” in the definition of SMPSA with “≤” or “=”? Anyway, these hints are mostly intended just to get you started thinking about the problem, and how to approach the question. There are many ways to write down a proof, and you are free to structure your solution any way you like. No matter how you organize your proof, your solution should start with a **clear outline** of how the rest of the proof is structured.]*

Problem 2: SPACE Complexity

Let $LINSPACE$ be the class of all problems that can be decided by a deterministic TM in linear space (more specifically $LINSPACE = \bigcup_c SPACE(c \cdot n)$). The goal of this problem is to show that $LINSPACE$ is different from NP .

1. Prove that NP is closed under polynomial time mapping reductions (namely if $A \leq_p B$ and $B \in NP$ then $A \in NP$).
2. Using the padding technique, demonstrate that there exist two languages $\mathcal{L}_1, \mathcal{L}_2$ such that
 - (a) $\mathcal{L}_1 \notin LINSPACE$,
 - (b) $\mathcal{L}_2 \in LINSPACE$, and
 - (c) $\mathcal{L}_1 \leq_p \mathcal{L}_2$.
3. Conclude that $LINSPACE \neq NP$, using the results proved in (1) and (2).

[Hint: The core of this problem is solving part (2). Questions (1) and (2) are independent, i.e., you can work on them separately, and in any order. When working on (2) recall that the space complexity is always computed as a function of the size of the input instance. Use the space hierarchy theorem to construct \mathcal{L}_1 . Question (3) depends on both (1) and (2), but you can work on it independently assuming (1) and (2) are given.]