Complexity Theory : Final exercise
(submission date 21/7/2009)

Teacher: Ronen Shaltiel

June 15, 2009

Instructions

General:

- You must hand in printed solutions. (I'm willing to accept calculations in handwriting if you don't know how to print them but the rest must be printed).
- Please write clearly and precisely!
- I will not accept solutions that are longer than 10 pages.

Rules:

- This is a test! You are not allowed to collaborate with other people and must do the work on your own. For some of the questions it may be the case that there are solutions on-line. Don't use them!
- Please include in your submission a **signed statement** saying that: I (*include name and ID*) hereby declare that I did not discuss this project with any other people and the solution that I am submitting is my own work. If you submit electronically (which is preferred) make sure to place such a statement in my mailbox. Note that I may want to set up a date in which I will spend 10-20 minutes with every student and have him explain his solutions to me.

Grading:

- There are 9 questions and you are supposed to answer only 3 of them. Each question i comes with two numbers (a_i, b_i). The final score will be given by summing the a_i's of the questions you answer to compute a, summing the b_i's of the questions you answer to compute b and the final score is $min(a, 70) + b$.
- Write clear and full answers! A 10 point bonus will be given to submissions which are clear and well written.
- You are allowed to answer one additional question over the 3 questions required (that is you can answer 4 questions), and I will base the scoring on the best 3 questions.

Good Luck!

Questions:

1. (25,0) (Formula minimization). Given a function $f : \{0,1\}^\ell \to \{0,1\}$ we can encode it as a string x_f of length $n = 2^\ell$ by considering its truth table. (That is $x_i = f(i)$). Consider the following language:

   $Min{-}Formula = \{x, s :$ The smallest boolean formula computing the function f encoded by x is of length $s\}$

   Show that $Min - Formula \in P^{NP}$.

2. (30,0) (st-connectivity in $O(1)$-regular graphs of logarithmic diameter) Design an algorithm that is given an undirected graph G on n vertices and two vertices s ant t. The algorithm should run in space $O(\log n)$ and output an answer in $\{0,1\}$ such that:

   - If the graph G does not have a path from s to t then the algorithm outputs 0.
   - If the graph G has a path from s to t that is of length at most $100 \log n$ and every vertex v in the graph has degree at most 100 then the algorithm outputs 1.

3. Consider the language:

   $$L = \{M, x, t : \text{deterministic TM } M \text{ accepts } x \text{ after at most } t \text{ steps}\}$$

   (a) (10,0) Show that $L \in EXP$.

   (b) (10,0) Show that L is complete for EXP in the sense that for every $L' \in EXP$ we have that $L' \leq_P L$.

   (c) (10,2) Show that $L \notin P$.

4. (Number of queries for NP^{NP}).

   (a) (15,3) Show that for every nondeterministic polynomial time oracle machine M there exists a nondeterministic polynomial time oracle machine N such that $L(N^{3-SAT}) = L(M^{3-SAT})$ and N makes a single query to its oracle.

   (b) (15,5) Show that every nondeterministic polynomial time oracle machine M and language $A \in RP$ there exists a nondeterministic polynomial time oracle machine N and a language $B \in RP$ such that $L(N^B) = L(M^A)$ and N makes a single query to its oracle.

5. (30,10) (zero sided error versus expected polynomial time)

   Definition 1. *A language L has a zero-sided error poly-time probabilistic algorithm if there exists a probabilistic poly-time algorithm A which gives outputs in $\{0, 1, ?\}$ and satisfies:*

   - *For every input x and coin-toss y if $A(x,y) \neq ?$ then $A(x,y) = 1_L(x)$.*
   - *For every input x, $\Pr_y[A(x,y) = ?] \leq 1/3$.*

   Definition 2. *A language L is in ZPP if there exists a probabilistic algorithm A such that for every input x and every coin toss y, $A(x,y) = 1_L(x)$ and there exists a polynomial p such that for every input x the expected running time of A on x is bounded by $p(|x|)$.*

   Show that a language L is in ZPP if and only if it has a zero-sided error poly-time probabilistic algorithm.

6. ($P/\log$). Recall that $P/\log$ is the class of all languages accepted by nonuniform Turing machines that receive advice of logarithmic length. More precisely, a language $L \in P/\log$ if there exist a polynomial time Turing machine and a sequence $\{\alpha_n\}_{n=1}^{\infty}$ of strings such that for every n, $|\alpha_n| = \log n$ and for every input x, $M(x, \alpha_{|x|}) = 1_L(x)$.

   (a) (15,0) Show that $P/\log \neq P$.

   (b) (15,10) Show that if $NP \subseteq P/\log$ then $NP = P$.

7. ($NTIME(n) \neq P$) We say that a class C of languages allows unpadding by a function t if whenever $L_t \in C$ then $L \in C$. (Here $L_t = \left\{1^{t(|x|)}0 \circ x : x \in L\right\}$ as in exercise 2).

   (a) (15,0) Show that P allows unpadding by any polynomial $t(n)$.

   (b) (20,12) Show that $NTIME(n)$ does not allow unpadding by $t(n) = n^5$.

   (c) Conclude that $NTIME(n) \neq P$. Did we just prove that $SAT \notin P$? (No need to answer).

8. (AM and perfect completeness). We first repeat the definition of the class AM being more precise with the completeness and soundness probabilities.

   Definition 3 (The class AM_s^c)**.** *A language L is in AM_s^c if there exists a polynomial time machine V and a constant c such that for every string x of length n:*

   - *If $x \in L$ then* $\Pr_{r \in_R \{0,1\}^{n^c}}[\exists a \in \{0,1\}^{n^c} : V(x,r,a) = 1] \geq c$.
   - *If $x \notin L$ then* $\Pr_{r \in_R \{0,1\}^{n^c}}[\exists a \in \{0,1\}^{n^c} : V(x,r,a) = 1] \leq s$.

   Recall that we defined $AM = AM_{1/3}^{2/3}$. Furthermore by parallel repetition one can show that $AM_{1/3}^{2/3} = AM_{1/2^{2n}}^{1-1/2^{2n}}$.

   (a) (25,15) Show that $AM_{1/3}^{2/3} = AM_{1/3}^{1}$. (Hint: use the techniques of the proof that $BPP \subseteq \Sigma_2^P$).

   (b) (10,5) Conclude that $AM \subseteq \Pi_2^P$.

9. (35,14) Show that: $PSPACE \subseteq P/poly \Rightarrow PSPACE = MA$. (Hint: it is easy to see that $MA \subseteq PSPACE$. For the other direction note that $PSPACE = IP$ and that we noted that the prover strategy in an IP protocol can be computed in $PSPACE$.)