Advanced Complexity Theory
6.841/18.405J - Spring 2009
Due: Wednesday, February 18, 2009

Instructor: Madhu Sudan
TA: Brendan Juba

Problem Set 1

Problems

1. **Non-deterministic time hierarchy:**
   Let $L_0 \in \text{NTIME}(n)$ be a unary language (i.e., $L \subseteq \{1\}^*$). Let $n_0 = 1$, $n_1 = 2, \ldots, n_{i+1} = 2^{n_i}$, ... denote the sequence of *towers* among the integers. Define L_1 as follows: If n is not a tower then $1^n \in L_1$ iff $1^{n+1} \in L_0$. If n is a tower then $1^n \in L_1$ iff $1^{\log_2 n} \notin L_0$.

   (a) Prove that $L_1 \neq L_0$.

   (b) Give the best function f that you can so that $L_1 \in \text{NTIME}(f(n))$.

   (c) Using the above give the best function g so that $\text{NTIME}(n) \subsetneq \text{NTIME}(g(n))$.

2. **Nondeterministic space-bounded computation:**
   The goal of this question is to ensure you can think about non-deterministic computation concretely.

   The language "PATH" is defined as

   $$\text{PATH} = \{(G, s, t) : G \text{ is a directed graph; } s, t \in G; \text{ there is a path from } s \text{ to } t \text{ in } G\}.$$

   (a) Write pseudo-code for a non-deterministic logspace (**NL**) algorithm for PATH.

   (b) Write pseudo-code for a non-deterministic logspace (**NL**) algorithm for co-PATH, the complement of PATH.

   Verify that the above algorithms indeed use logarithmic space. (To think concretely about this question, try to add "non-deterministic" primitives to your pseudo-coding language. Useful primitives would be "Guess $x \in S$" for some finite set S, and "If P then ACCEPT" or "If P then CONTINUE" for some predicate P. P itself maybe computed by some non-deterministic computation.)

3. **Space-efficient Boolean matrix multiplication and consequences:**
   Given two $n \times n$ matrices A, B with Boolean entries, their boolean product $A \cdot B$ is the matrix C such that

   $$C_{ij} = \bigvee_{k=1}^{n} (a_{ik} \wedge b_{kj})$$

   (a) Give a Logspace algorithm to compute $A \cdot B$ given A and B. (Food for thought: How can the algorithm take less space than the output length?)

(b) Given matrix A and integer k, give a small space algorithm to compute A^k the kth Boolean power of A. How much space does your algorithm use? What well-known theorem follows from this algorithm?

4. **Ladner's general theorem:**
Let L_1, L_2 be languages such that L_1 is polynomial time reducible to L_2 (denoted $L_1 \leq_P L_2$), but L_2 is not polynomial time reducible to L_1. Show that there exist 2 languages L_a and L_b such that:
   - $L_1 \leq_P L_a \leq_P L_2$
   - $L_1 \leq_P L_b \leq_P L_2$
   - $L_a \not\leq_P L_b$
   - $L_b \not\leq_P L_a$

5. **Approximation and Inapproximability:**
The goal of this question is principally to test your ability to pose and use decision problems to capture computational complexity. A secondary goal is to remind you about NP-completeness.

   (a) The input to the ASYMMETRIC k-CENTER problem is a directed graph $G = (V, E)$ and an integer k. The output should be a subset S containing at most k vertices of G that minimimzes the quantity $\max_{x \in V}\{\min_{y \in S}\{d(x, y)\}\}$, where $d(x, y)$ denotes the length of the shortest path from x to y in G. (The graph may be assumed to be weighted/unweighted depending on your preference.)

   Show that the ASYMMETRIC k-CENTER problem is NP-hard to solve by posing an appropriate decision problem, and showing this decision problem to be NP-complete. (Hint: Reduce from Vertex Cover.)

   (b) Let $\mathrm{Obj}(S)$ denote the quantity $\max_{x \in V}\{\min_{y \in S}\{d(x, y)\}\}$. An α-approximation algorithm for ASYMMETRIC k-CENTER is a polynomial time algorithm that, given (G, k), outputs a set S with $|S| = k$ such that for every $S' \subseteq V$ with $|S'| = k$, it is the case that $\mathrm{Obj}(S) \leq \alpha\, \mathrm{Obj}(S')$.

   Show that there exists some $\alpha > 1$ for which an α-approximation algorithm for the ASYMMETRIC k-CENTER problem would imply NP=P. (The larger the α the better.)

   (c) A promise problem is a class of "Boolean" computational problems given by a pair of *disjoint* sets of instances $\Pi = (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$. (A standard decision problem is simply the special case where $\Pi_{\mathrm{NO}} = \overline{\Pi_{\mathrm{YES}}}$. An algorithm A decides a promise problem Π if for every $x \in \Pi_{\mathrm{YES}}$, $A(x) = 1$ and for every $x \in \Pi_{\mathrm{NO}}$, $A(x) = 0$. An algorithm R reduces a promise problem Π to a promise problem Γ if $x \in \Pi_{\mathrm{YES}}$ implies $R(x) \in \Gamma_{\mathrm{YES}}$ and $x \in \Pi_{\mathrm{NO}}$ implies $R(x) \in \Gamma_{\mathrm{NO}}$.

   Given an integer c, describe a promise problem Π related to the ASYMMETRIC k-CENTER problem such that the existence of a polynomial time reduction from Vertex Cover to your problem Π would rule out the existence of a c-approximation algorithm for the ASYMMETRIC k-CENTER problem unless NP=P. (So you have to describe Π_{YES} and Π_{NO}. You don't have to reduce Vertex Cover to this promise problem. What you have to show is that if you assume such a reduction and also a c-approximation algorithm for the ASYMMETRIC k-CENTER problem, you get NP=P.)