198:538 Complexity of Computation
Rutgers University, Spring 2007

Homework 4
due Tuesday 10 April

Instance checkers Suppose you have an algorithm that solves some difficult problem, but you are not sure if the algorithm is correct. Ideally, you would want to first check the correctness of the algorithm, then use it to solve the instance at hand. An instance checker is a procedure that combines both these steps into one.

An *instance checker* for a decision problem L is a randomized polynomial-time algorithm I that has oracle access to a candidate algorithm A for L and has the following behavior:

- For every A and every x, $I^A(x)$ outputs one of 0 (”reject”), 1 (”accept”), or ”fail”.
- If $A(x) = L(x)$ for every x, then $I^A(x) = L(x)$ for every x. (If A is a good algorithm for L, then I^A always outputs the correct answer.)
- For every A and every x, $\Pr[I^A(x) \in \{L(x), \text{”fail”}\}] \geq 3/4$. (If A is not a good algorithm for L, then with high probability I^A either fails or outputs the correct answer.)

Similarly we can define instance checkers for search problems and counting problems.

Problem 1

In $\mathbb{F}_2$-matrix multiplication you are given two $n \times n$ matrices A, B over $\mathbb{F}_2$ and want to compute the matrix product AB. The best known algorithm for matrix multiplication takes time $\Omega(n^{2.3})$. Show that matrix multiplication has an instance checker that runs in time $\tilde{O}(n^2)$. (Assume you have random access to the input.)

Hint: Check the answer by multiplying it with a random vector.

Problem 2

In this problem you investigate instance checkers for PSPACE-complete problems.

(a) Show that every PSPACE-complete decision problem has an instance checker.

Hint: Look at the interactive proof for PSPACE. The prover in this protocol can be realized by a polynomial space machine. Let I play the role of the verifier and A play the role of the prover.

(b) Let L be a PSPACE-complete decision problem. Show that there is a randomized algorithm A for L with the following property. For every (not necessarily efficient) randomized algorithm M that decides L and every input x, if $M(x)$ halts within t steps, then $A(x)$ halts within $p_M(|x|, t)$ steps, where p_M is some polynomial whose coefficients may depend on the description of M but not on x or t.

Hint: Your algorithm A should run the instance checker for L, using candidate algorithms M as the oracle.

Problem 3

In class we saw that the Hadamard code is locally testable. Local testability also applies to other codes. Let's fix a finite field $\mathbb{F}$ and parameters d, n, where $d < |\mathbb{F}|$. The Reed-Muller code of degree d encodes every polynomial p of degree d in n variables by the vector $RM_p \in \mathbb{F}^{|\mathbb{F}|^n}$ where $RM_p(x_1, \ldots, x_n)$ is the value of the polynomial p at the point $(x_1, \ldots, x_n) \in \mathbb{F}^n$.

The Reed-Muller code is locally testable: There exists a probabilistic algorithm A with oracle access to a candidate encoding f that runs in time $\text{poly}(n, d, |\mathbb{F}|)$, makes $\text{poly}(d, |\mathbb{F}|)$ queries to the oracle, and if $\Pr[A^f \text{ accepts}] \geq 7/8$, then f is 7/8-close to RM_p for some polynomial p of degree d.

(a) Argue that the family of permanent polynomials is the unique family of polynomials $p_1, p_2, \ldots$ that satisfies the system of equations

$$p_n(x_{ij})_{1 \leq i,j \leq n} = \sum_{k=1}^{n} x_{1k} \cdot p_{n-1}(x_{ij})_{1 \leq i,j \leq n, i \neq 1, j \neq k}$$
$$p_{n-1}(x_{ij})_{1 \leq i,j \leq n-1} = p_n(y_{ij})_{1 \leq i,j \leq n}$$

where

$$y_{ij} = \begin{cases} x_{ij}, & \text{if } 1 \leq i, j \leq n-1, \\ 1, & \text{if } i = j = n, \\ 0, & \text{otherwise.} \end{cases}$$

and $p_1(x_{11}) = x_{11}$.

(b) Using the local testability of the Reed-Muller code, show that the problem of computing the permanent of an $n \times n$ matrix over fields $\mathbb{F}$ of size $\Theta(n^2)$ has an instance checker.

Hint: Think of a good oracle as providing the Reed-Muller encoding of the permanent polynomial. In addition to the fact that the Reed-Muller code is locally testable, you will need to use the reconstruction algorithm for the permanent and the randomized algorithm for polynomial identity testing.