

Final Exam - Algorithmic Algebra.

May 10, 2008

NOTE: The answers are due by 10 AM on Monday, 12th May 2008.

Problems

1. (15 points.) **Radicals of matrices.** In this exercise we will devise an algorithm that given a matrix A computes its k -th radical, $A^{\frac{1}{k}}$. More precisely, the task is to devise an efficient randomized algorithm that given a prime p , an integer $k \geq 2$ and an $(n \times n)$ -matrix $A \in \mathbb{F}_p^{n \times n}$, computes a matrix $B \in \mathbb{F}_p^{n \times n}$ such that $B^k = A$, if such a matrix B exists.

Notes:

- The running time of the algorithm should be polynomial in $(\log p \cdot n \cdot k)$.
- You may assume that $k < p$.

2. (15 points.) **Discrete Logarithms for matrices.**

Background: The discrete logarithm problem (for elements of a field) is the following: Given a finite field \mathbb{F}_q and two elements $\alpha, \beta \in \mathbb{F}_q$, the task is to compute the smallest integer $m \geq 0$ such that $\alpha^m = \beta$, if such an m exists. No efficient algorithm is known for this problem and indeed, there are many cryptosystems based on the presumed difficulty of solving this problem.

The problem at hand. A new cryptosystem has been designed whose security is based on the assumption that the discrete logarithm problem for matrices (to be precisely defined shortly) is hard, possibly harder than solving discrete logarithm over finite fields. Your task is to show that the new cryptosystem is no more secure than the known cryptosystems by showing that the discrete logarithm problem for matrices is no harder than the usual discrete logarithm problem for field elements. More precisely, the task is to devise a randomized polynomial-time algorithm for solving discrete logarithm for matrices assuming that we have an oracle for solving the usual discrete logarithm problem for field elements.

The discrete logarithm problem for matrices is the following: given a finite field \mathbb{F}_p and two $(n \times n)$ matrices $A, B \in \mathbb{F}_p^{n \times n}$, find the smallest integer $m \geq 0$ such that $A^m = B$, if such an m exists.

3. (10 points.) **Computing square roots deterministically.** The problem of computing square roots modulo a prime is the following: given a prime p and an integer $a \in \mathbb{F}_p$, the task is to compute an integer $b \in \mathbb{F}_p$ such that $b^2 = a \pmod{p}$. In this exercise we will see that if we are given a certain root of unity in the finite field \mathbb{F}_p then we can compute all square roots efficiently.

Recall that a primitive m -th root of unity is an element ω such that m is the smallest integer so that $\omega^m = 1$. Show that:

- (a) (3 points) There exists a primitive m -th root of unity in \mathbb{F}_p if and only if m divides $(p - 1)$.
 - (b) (5 points) Let $p - 1 = 2^\ell \cdot s$, where s is an odd integer. Show that if in addition to the prime p , we are also given a primitive 2^ℓ -th root of unity $\omega \in \mathbb{F}_p$, we can compute the square-root of *any* element in \mathbb{F}_p in *deterministic* polynomial time.
 - (c) (2 points) Show that if $\alpha \in \mathbb{F}_p$ is a quadratic nonresidue then $\alpha^{\frac{p-1}{2^\ell}}$ is a primitive 2^ℓ -th root of unity in \mathbb{F}_p .
4. (20 points.) **Computing the elementary symmetric polynomials.** Recall that the d -th elementary symmetric polynomial in n variables x_1, x_2, \dots, x_n is the following polynomial:

$$S_d(x_1, x_2, \dots, x_n) \stackrel{\text{def}}{=} \sum_{1 \leq i_1 < i_2 < \dots < i_d \leq n} x_{i_1} \cdot x_{i_2} \cdot \dots \cdot x_{i_d}.$$

- (a) (5 points.) Devise a $\text{poly}(n \cdot d)$ -sized arithmetic circuit that computes $S_d(\mathbf{x})$.
- (b) (15 points.) We will now consider unbounded fanin arithmetic circuits of depth three. Such a circuit C computes a polynomial of the form

$$C(\mathbf{x}) = \sum_{i \leq T} \prod_{j \leq D} L_{ij}(\mathbf{x}),$$

where the $L_{ij}(\mathbf{x})$'s are polynomials of degree one. That is, every $L_{ij}(\mathbf{x})$ is of the form

$$L_{ij}(\mathbf{x}) = a_0 + \sum_{1 \leq k \leq n} a_k \cdot x_k.$$

For such a circuit C , its size is $(T \cdot D \cdot n)$.

Now devise a $\text{poly}(n \cdot d)$ -sized depth three circuit that computes $S_d(\mathbf{x})$.

5. (5 points) Prove or disprove the following conjecture.

Let $f(x, y), g(x, y) \in \mathbb{C}[x, y]$ be any two polynomials. Let I_{-x} be the elimination ideal of f and g for the variable x . That is,

$$I_{-x} \stackrel{\text{def}}{=} \langle f(x, y), g(x, y) \rangle \cap \mathbb{C}[y].$$

Let $\rho(y) \stackrel{\text{def}}{=} \text{RES}_x(f(x, y), g(x, y))$. Then $I_{-x} = \langle \rho(y) \rangle$. In other words, the conjecture states that every polynomial in the elimination ideal I_{-x} is a multiple of the resultant polynomial $\rho(y)$.