Assignment 4

Due date: April 24, 2012

General instructions:

- Submit your solutions by typesetting in LaTeX.
- Write your solutions by furnishing all relevant details (you may assume the results already covered in the class or previous homework problems).
- You are strongly urged to solve the problems by yourself.
- If you discuss with someone else or refer to any material (other than the course notes) then please put a reference in your answer script stating clearly whom or what you have consulted with and how it has benifited you. We would appreciate your honesty.
- If you need any clarification, please ask one of the instructors.

Total: 55 points

1. (**5 points**) Let $\mathbb{F}$ be a field and $\mathbf{x} \stackrel{\text{def}}{=} (x_1, \ldots, x_n)$ be an n-tuple of indeterminates. A set of nonzero polynomials $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ is said to be $\mathbb{F}$-linearly dependent if there exist constants $a_1, \ldots, a_m \in \mathbb{F}$, not all zero, such that
$$a_1 \cdot f_1(\mathbf{x}) + a_2 \cdot f_2(\mathbf{x}) + \ldots + a_m \cdot f_m(\mathbf{x}) = 0.$$
Suppose that every $f_i(\mathbf{x})$ is given in the form of an arithmetic circuit with size bounded by s. Devise a randomized algorithm that given polynomials $f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})$ in the form of arithmetic circuits, tests whether $f_1, \ldots, f_m$ are $\mathbb{F}$-linearly dependent, in time polynomial in s.

2. (**5 points**) In this exercise you will devise an algorithm that given a matrix A computes its k-th radical, $A^{1/k}$. 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. The running time of the algorithm should polynomial in $\log p$, n and k. You can assume that all the eigenvalues of A are in $\mathbb{F}_p$ and $p > \max\{k, n\}$. [Hint: If it helps, you can also assume that the Jordan normal form of A can be computed in randomized polynomial time i.e. an invertible matrix $C \in \mathbb{F}_p^{n \times n}$ can be computed in (randomized) time polynomial in n and $\log p$ such that CAC^{-1} is in Jordan normal form.]

3. (**5 points**) The discrete logarithm problem for matrices is the following: given a prime 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. (You can assume that we are given the promise that m, if it exists, is less than p). Your task is to show that the discrete logarithm problem for matrices is no harder than the usual discrete log problem for field elements (1-dimensional matrices over $\mathbb{F}_p$). More precisely, devise a randomized polynomial time algorithm for solving the discrete log problem for matrices assuming that we have an oracle for solving the usual discrete log problem for $\mathbb{F}_p$ elements. Your algorithm should run in time polynomial in n and $\log p$. [Hint: You may assume the hint given in Problem 2.]

4. (**5 points**) Consider the following polynomial factorization task: Given as input a prime p and polynomial $f \in \mathbb{F}_p[x]$ of degree n that splits completely over $\mathbb{F}_p$, design a deterministic algorithm, by adapting the Pollard-Strassen method, that factors f using $\tilde{O}(n\sqrt{p})$ operations in $\mathbb{F}_p$.

5. (**15 points**) In the class, we have seen how a multivariate polynomial f can be factored by quering a black-box for evaluations of f followed by sparse polynomial interpolation. However, for the interpolation algorithm to work, we need the knowledge of the sparsity of every factor of f. In this exercise, we will see how to efficiently find the sparsity of a polynomial g from its evaluations at several points.

   Let $g(\mathbf{x}) \in \mathbb{F}_q[\mathbf{x}]$ be a polynomial in n variables $\mathbf{x} = (x_1, \ldots, x_n)$ with total degree bounded by d. Suppose, g has k monomials i.e. $g = \sum_{i=1}^{k} a_i M_i(\mathbf{x})$, where $M_i(\mathbf{x})$ is a monomial, and $a_i \in \mathbb{F}_q \backslash \{0\}$ for every $1 \leq i \leq k$. Consider the following sparsity test which checks if $k \leq m$. Define, $\mathbf{x}^i = (x_1^i, \ldots, x_n^i)$.

1 Sparsity test

1. Pick $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$ uniformly at random.
2. Let
$$H_{m+1}(\mathbf{x}) = \begin{pmatrix} g(\mathbf{x}^0) & g(\mathbf{x}^1) & \ldots & g(\mathbf{x}^m) \\ g(\mathbf{x}^1) & g(\mathbf{x}^2) & \ldots & g(\mathbf{x}^{m+1}) \\ \vdots & \vdots & \ldots & \vdots \\ g(\mathbf{x}^m) & g(\mathbf{x}^{m+1}) & \ldots & g(\mathbf{x}^{2m}) \end{pmatrix}_{(m+1)\times(m+1)}.$$
   Compute $\det(H_{m+1}(\mathbf{x}))$.
3. If $\det(H_{m+1}(\mathbf{x})) = 0$, output '$g$ is m-sparse'.
4. Else, output 'g is not m-sparse'.

We would like to prove the following theorem.

Theorem 0.1 *Let $q \geq \frac{m(m+1)d}{\epsilon}$. Then the above algorithm outputs 'g is m-sparse' if $k \leq m$, and outputs 'g is not m-sparse' with probability at least $1 - \epsilon$ if $k > m$.*

(a) (**2 points**) Show that $H_{m+1}(\mathbf{x})$ equals the following product for any $m \in \mathbb{N}$.

$$\begin{pmatrix} 1 & 1 & \ldots & 1 \\ M_1(\mathbf{x}) & M_2(\mathbf{x}) & \ldots & M_k(\mathbf{x}) \\ M_1(\mathbf{x}^2) & M_2(\mathbf{x}^2) & \ldots & M_k(\mathbf{x}^2) \\ \vdots & \vdots & \ldots & \vdots \\ M_1(\mathbf{x}^m) & M_2(\mathbf{x}^m) & \ldots & M_k(\mathbf{x}^m) \end{pmatrix} \times \begin{pmatrix} a_1 & 0 & \ldots & 0 \\ 0 & a_2 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & a_k \end{pmatrix} \times \begin{pmatrix} 1 & M_1(\mathbf{x}) & \ldots & M_1(\mathbf{x}^m) \\ 1 & M_2(\mathbf{x}) & \ldots & M_2(\mathbf{x}^m) \\ 1 & M_3(\mathbf{x}) & \ldots & M_3(\mathbf{x}^m) \\ \vdots & \vdots & \ldots & \vdots \\ 1 & M_k(\mathbf{x}) & \ldots & M_k(\mathbf{x}^m) \end{pmatrix}$$

(b) (**3 points**) Use (a) to infer that, if $k \leq m$ then $\det(H_{m+1}(\mathbf{x})) = 0$ as a polynomial in $x_1, \ldots, x_n$.

(c) (**5 points**) Use (a) to show that, if $k > m$ then the following expression holds:

$$\det(H_{m+1}(\mathbf{x})) = \sum_{S \subset [k], |S| = m+1} \prod_{i \in S} a_i \cdot \prod_{i<j, j \in S} (M_i(\mathbf{x}) - M_j(\mathbf{x}))^2$$

(d) (**3 points**) Show that the RHS of the above equation in (c) is a *non-zero* polynomial in $x_1, \ldots, x_n$ with total degree bounded by $m(m+1)d$.

(e) (**2 points**) Infer that in the above algorithm $\det(H_{m+1}(\mathbf{x})) \neq 0$ with probability at least $1 - \epsilon$ if $k > m$ and $q > \frac{m(m+1)d}{\epsilon}$.

You can infer from here that by doing a 'binary search' on m, we can find the value of k.

6. (**10 points**) The discrete logarithm problem over a prime field $\mathbb{F}_p$ is the following: Given $a, b \in \mathbb{F}_p^\times$, compute an x such that $a^x = b \mod p$ if such an x exists. Show that, if $p - 1$ is an ℓ-smooth number then the discrete logarithm problem over $\mathbb{F}_p$ can be solved deterministically in time polynomial in ℓ and $\log p$.

7. (**10 points**) The Chevalley-Warning-Ax theorem states the following: The number of roots of a degree d polynomial $f(x_1, \ldots, x_n) \in \mathbb{F}_p[\mathbf{x}]$ is either 0 or it is at least p^{n-d}, provided $n > d$. Here, by roots we mean points in $\mathbb{F}_p^n$ at which f vanish modulo p. (p is a prime.)

   Use the above theorem to show that the number of boolean roots of $f(x_1, \ldots, x_n)$ is either 0 or it is at least $2^{n-d(p-1)\log p}$, assuming $n > d(p-1)$. Here, by a boolean root we mean a point $(a_1, \ldots, a_n) \in \{0,1\}^n$ such that $f(a_1, \ldots, a_n) = 0 \mod p$.

   [Hint: Use probabilistic method. For a "random" vector $r = (r_1, \ldots, r_n) \in \{0,1\}^n$, how can you connect the roots of $f_r(x_1, \ldots, x_n)$ defined as,

$$f_r(x_1, \ldots, x_n) = f(r_1 x_1^{p-1} + (1 - r_1)(1 - x_1^{p-1}), \ldots, r_n x_n^{p-1} + (1 - r_n)(1 - x_n^{p-1}))$$

   with boolean roots of f?]