Foundations of Computer Algebra: Assignment 1

Marc Moreno Maza
University of Western Ontario
CS424 / CS556 – Winter 2006

Posted: Friday, January 31, 2006
Due: Monday, February 14, 2006

1 Guidelines

Date of postage. Thursday, January 31, 2008.

Normal date for submission. Thursday, February 14, 2008.

Due format. The assignment can be returned to the instructor by email in a form of a postscript, PDF or text file. Alternatively, a printed version can be delivered to his mail box in the Computer Science Department or given to him in class.

Collaboration. You are expected to do this assignment on your own without assistance from anyone else in the class. However, you can use the literature, and in particular the book *Modern Computer Algebra* and the references therein.

Marking. This assignment will be marked out of 40. Each of the Exercises 1 and 2 is worth 12 points. Each of the Exercises 3 and 4 is worth 8 points.

2 Exercise 1

2.1 Statement

Let $\mathbb{K}$ be a field and $f \in \mathbb{K}[x]$ be a univariate polynomial of positive degree n. We denote by $\mathbb{L}$ the residue class ring $\mathbb{K}[x]/\langle f \rangle$. Since we do not assume that f is an irreducible polynomial, some non-zero elements in $\mathbb{L}$ may not have an inverse. This motivates the following notion. For any $a \in \mathbb{L}$, with $a \neq 0$, we call *quasi-inverse* of a, any $b \in \mathbb{L}$, with $b \neq 0$, such that either $ab = 0$ or $ab = 1$ holds.

2.2 Questions

(1) Explain why the elements of $\mathbb{L}$ can be encoded as vectors with n coordinates in $\mathbb{K}$. In the remaining questions, we will assume that this encoding is used.

(2) When does an element of $\mathbb{L}$ have an inverse in $\mathbb{L}$?

(3) With $\mathbb{K} = \mathbb{Z}/p\mathbb{Z}$, for some prime integer p, and $f = x^2+3x+2$, compute a quasi-inverse of $a_1 = x + 1$ and $a_2 = x - 1$.

(4) Show that every $a \in \mathbb{L}$, with $a \neq 0$, possesses a quasi-inverse.

(5) Show that a quasi-inverse of $a \in \mathbb{L}$, with $a \neq 0$, can be computed in $O(n^2)$ operations in $\mathbb{K}$.

A quasi-inverse of $a \in \mathbb{L}$, with $a \neq 0$, tells us whether a is invertible or a zero-divisor. In practice, the second case is inconvenient. For instance, division by a zero-divisor is not uniquely defined. This motivates the last questions of this exercise. From now on, we denote by a a non-zero polynomial in $\mathbb{K}[x]$ with degree less than n.

(6) Using GCD computations in $\mathbb{K}[x]$, show that one can compute polynomials $f_0, \ldots, f_e$ such that their product is f and such that for each $i = 0 \cdots e$ the polynomial a is either zero or invertible modulo f_i.

Analyzing the complexity for computing $f_1, \ldots, f_e$ in question (6) is not easy and not required. However, one special case is simpler: when f is squarefree, that is, when for all non-constant polynomial $h \in \mathbb{K}[x]$, the polynomial h^2 does not divide f. From now on, we assume that f is squarefree.

(7) Using GCD computations in $\mathbb{K}[x]$, show that one can compute polynomials $f_0, f_1 \in \mathbb{K}[x]$ such that $f = f_0 f_1$, and a is null modulo f_0, and a is invertible modulo f_1. Show that f_0, f_1 and the inverse of a modulo f_1 can be computed in $O(n^2)$ operations in $\mathbb{K}$.

3 Exercise 2

3.1 Statement

We saw in class that Karatsuba's trick can reduce the cost of multiplying two univariate polynomials. This trick splits each input polynomial in two parts: its low degree terms and high degree terms. It is natural to ask whether there is a similar trick

- which splits each input polynomial into three parts, and
- which leads to a time complexity improvement in computing their product.

To make things simple, let us restrict ourselves to two polynomials $A = a_n X^n + \cdots + a_0$ and $B = b_n X^n + \cdots + b_0$ with the same degree n and let us assume that n is a power of 3.

3.2 Questions

(1) Find formulas for a three parts Karatsuba's trick. Using MAPLE or AXIOM, for setting up these formulas, would be a good idea.

(2) Is this three parts Karatsuba's trick theoretically better than the usual two parts one? That is, does this lead to a better time complexity upper bound?

You will need the so-called *Master Theorem* for analyzing the complexity of divide-and-conquer algorithms. See the document *Master Theorem: Practice Problems and Solutions* posted on our web site.

4 Exercise 3

4.1 Statement

Let a and b be two univariate polynomials in $\mathbb{K}[x]$, with respective degrees m and $n > 0$. We call *b-adic expansion* of a, a sequence of polynomials $c_k, \ldots, c_1, c_0$ in $\mathbb{K}[x]$ such that

1. $a = c_k b^k + \cdots + c_1 b + c_0$,
2. the degree of each $c_k, \ldots, c_1, c_0$ is strictly less than that of b,
3. $c_k = 0$ holds if and only a is null.

This is an adaptation to polynomials of a well-known notion for integer numbers.

4.2 Questions

(1) Consider $a = x^2 + x + 1$ and $b = x - 1$. Using the Euclidean division, compute a b-adic expansion of a

(2) For the general case, design an algorithm that computes a b-adic expansion of a.

(3) Give an upper bound for the time complexity of this algorithm, counting the number of operations in $\mathbb{K}$.

(4) Show that a admits one and only one b-adic expansion.

5 Exercise 4

5.1 Statement

Let $p \in \mathbb{N}$ and $n = 2^p$. Let A and B be two square matrices of order n and with coefficients in $\mathbb{Z}$. The classical algorithm for computing the matrix product $A\,B$ uses $\mathcal{O}(n^3)$ operations in $\mathbb{Z}$. The *Strassen* algorithm provides the lower asymptotic upper bound $\mathcal{O}(n^{2,81})$. As for the *Karatsuba* algorithm, the trick relies on a smart use of distributivity:

(1) If $n = 1$ the matrices A et B are of the form (a) et (b) where $a, b \in \mathbb{Z}$, such that $A\,B$ equals $(a\,b)$.

(2) If $n > 1$ the matrices A and B can be decomposed as

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix},\ B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix},$$

where A_{ij} and B_{ij} are square matrices of order $n/2$. Then, one computes the following 8 sums.

$$\begin{array}{ll} S_1 = A_{21} + A_{22} & T_1 = B_{12} - B_{11} \\ S_2 = S_1 - A_{11} & T_2 = B_{22} - T_1 \\ S_3 = A_{11} - A_{21} & T_3 = B_{22} - B_{12} \\ S_4 = A_{12} - S_2 & T_4 = T_2 - B_{21} \end{array}$$

(3) Next, one computes the following 7 matrix products by recursive calls to the algorithm.

$$\begin{array}{ll} P_1 = A_{11}\,B_{11} & P_5 = S_1\,T_1 \\ P_2 = A_{12}\,B_{21} & P_6 = S_2\,T_2 \\ P_3 = S_4\,B_{22} & P_7 = S_3\,T_3 \\ P_4 = A_{22}\,T_4 & \end{array}$$

(4) Finally, one computes the following 7 sums:

$$\begin{array}{ll} U_1 = P_1 + P_2 & U_5 = U_4 + P_3 \\ U_2 = P_1 + P_6 & U_6 = U_3 - P_4 \\ U_3 = U_2 + P_7 & U_7 = U_3 + P_5 \\ U_4 = U_2 + P_5 & \end{array}$$

and one returns:

$$\begin{pmatrix} U_1 & U_5 \\ U_6 & U_7 \end{pmatrix}.$$

5.2 Questions

(1) Using AXIOM or Maple check that the above formulas are correct. Your answer must be an AXIOM input file or a Maple worksheet.

(2) Use the results of the first chapter to retrieve the asymptotic upper bound given above.