Foundations of Computer Algebra: Assignment 3

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

Posted: Thursday, March 2, 2006
Due: Sunday, March 26, 2006

1 Guidelines

Date of postage. Thursday, March 2, 2006.

Normal date for submission. Sunday, March 26, 2006.

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 exercise is worth 10 points.

2 Exercise 1

2.1 Statement

Let $p > 2$ be a prime integer. Let f and g in $\mathbb{Z}/p\mathbb{Z}[x]$ be two polynomials with respective degrees $m \geq n > 0$. We aim at computing the product $f\,g$ by a fast algorithm. We consider here the case where $m+n+1$ does not divide $p-1$. Hence, there are no $(n+m+1)$-th primitive roots of unity in $\mathbb{Z}/p\mathbb{Z}$ and the FFT-based algorithm studied in class does not apply in this context. However, we assume that we can find r primes numbers $2 < p_1 < \cdots < p_r$ such that:

- the product m of these primes exceeds $C = (n+1)(\frac{p-1}{2})^2$,
- every prime $p_1, \ldots, p_r$ fits in a machine word,
- for all $i = 1, \ldots, r$, the integer $m+n+1$ divides $p_i - 1$, such that, regarding f, g as polynomials in $\mathbb{Z}/p_i\mathbb{Z}[x]$, one can compute the product fg in $\mathbb{Z}/p_i\mathbb{Z}[x]$ using the FFT-based algorithm studied in class.

Based on this hypothesis, we are to design a modular algorithm for computing fg in $\mathbb{Z}/p\mathbb{Z}[x]$.

2.2 Questions

(1) Show that C is an upper bound for the coefficients of $f\,g$ computed in $\mathbb{Z}[x]$ (when regarding f and g as polynomials of $\mathbb{Z}[x]$).

(2) Describe a modular algorithm for computing $f\,g$ in $\mathbb{Z}[x]$ **and** give an upper for the number of machine word operations required by this algorithm.

(3) Deduce an algorithm for computing $f\,g$ in $\mathbb{Z}/p\mathbb{Z}[x]$ **and** give an upper for the number of machine word operations required by this algorithm.

2.3 Hints

One can reuse the results established in Exercise 2 of Assignment 2. For Question (3), you will need to take into account the cost of reducing modulo p the product $f\,g$ computed in $\mathbb{Z}[x]$, which means taking into account the cost of reducing each of these coefficients modulo p.

3 Exercise 2

3.1 Statement

Let $\mathbb{R}$ be a commutative ring and let $n > 0$ be a power of 2. Let $f = \Sigma_{k=0}^{k=n-1} f_k x^k$ and $g = \Sigma_{k=0}^{k=2n-1} g_k x^k$ be two polynomials in $\mathbb{R}[x]$ with degrees less than n and $2n$ respectively. The product $h = fg$ has degree less than $3n$ and we can write

$$h = a_0 + \cdots + a_{n-1}x^{n-1} + b_n x^n + \cdots + b_{2n-1}x^{2n-1} + c_{2n}x^{2n} + \cdots c_{3n-1}x^{3n-1},$$

where $a_0, \ldots, a_{n-1}, b_n, \ldots, b_{2n-1}, c_{2n}, \ldots, c_{3n-1}$ belong to $\mathbb{R}$. The goal of this exercise is to show that one can compute the coefficients $b_n, \ldots, b_{2n-1}$ at the cost of multiplying two polynomials in $\mathbb{R}[x]$ with degree $2n-1$. The polynomial $b_n x^n + \cdots + b_{2n-1}x^{2n-1}$ is called the *middle product* of f and g and plays an important role as we shall see in class.

3.2 Questions

(1) Show that the middle product of f and g can easily be obtained from the remainder of h by $x^{2n} - 1$.

(2) Let ω be a primitive $2n$-th root of unity. Deduce an algorithm that computes the middle product of f and g at the cost of multiplying two polynomials in $\mathbb{R}[x]$ with degree $2n-1$ by means of the FFT-based algorithm studied in class.

4 Exercise 3

4.1 Statement

Let $\mathbb{R}$ be a commutative ring and let $n > 0$ be a power of 2. Let ω be a primitive $2n$-th root of unity. We consider the following polynomial $m = x^n - x - 1$ in $\mathbb{R}[x]$. Let $f = \Sigma_{k=0}^{k=n-1} f_k x^k$ and $g = \Sigma_{k=0}^{k=n-1} g_k x^k$ be two polynomials of degree less than n. The goal of this exercise is to show that one can compute $fg \mod m$ *asymptotically* at the cost of multiplying two polynomials in $\mathbb{R}[x]$ with degree $n-1$. Let q and r be the quotient and the remainder in the division of fg by m. We define the polynomial $c = -qx + r$.

4.2 Questions

(1) Show that c can be computed in $\frac{9}{2}n\log n + O(n)$ operations in $\mathbb{R}$.

(2) Reusing the results of the intermediate calculations of (1), show that no more than $\frac{3}{2}n\log n + O(n)$ operations in $\mathbb{R}$ are needed for computing q.

(3) Deduce that r can be computed in $6n\log n + O(n)$ operations in $\mathbb{R}$.

(4) Give a sharp estimate for the *naive approach*, that is for:

   1. computing the product fg in $\mathbb{R}[x]$ and then
   2. computing the division of fg by m.

5 Exercise 4

5.1 Statement

Let $\mathbb{R}$ be a commutative ring. Let $d \longmapsto M(d)$ be a *multiplication time*. Hence, for any two polynomials $f, g \in \mathbb{R}[x]$ of degree less than d one can compute the product fg in at most $M(d)$ operations in $\mathbb{R}$. Moreover, for all $d' \geq d \geq 0$ we have $M(d') + M(d) \leq M(d' + d)$.

Let $d' \geq d > 0$ be two integers. Let $f \in \mathbb{R}[x]$ be of degree d' and let $m_1, \ldots, m_r \in \mathbb{R}[x]$ be monic polynomials such that the sum of their degrees $\deg m_1 + \cdots + \deg m_r$ is d.

The goal of this exercise is to show that all remainders $f \mod m_1, \ldots, f \mod m_r$ can be computed in $O(M(d)\log d + M(d'))$ operations in $\mathbb{R}$.

The following result (to be proved in class) will be essential: division-with-remainder of a polynomial $a \in \mathbb{R}[x]$ of degree $n + m$ by a monic polynomial $b \in \mathbb{R}[x]$ of degree n (where n, m are non-negative integers) can be done in at most $D(m, n)$ ring operations in $\mathbb{R}$ where we have

$$D(m,n) = \begin{cases} 4M(m) + M(n) + O(n) & \text{if} \quad m \leq n, \\ 5M(m) + O(m) & \text{if} \quad m \geq n. \end{cases}$$

5.2 Questions

(1) Consider the particular case where $d' = 4\,d$, $r = 2$ and $M(d) = 2d^2$. Compare the costs of the following two approaches:

1. compute first the remainder $r_{12} = f \mod m_1 m_2$ and then, the remainders $r_1 = r_{12} \mod m_1$ and $r_2 = r_{12} \mod m_2$,
2. compute the remainders $f \mod m_1$ and $f \mod m_2$ directly.

(2) Using a divide-and-conquer strategy, show that all remainders $f \mod m_1, \ldots, f \mod m_r$ can be computed in $O(M(d) \log d)$ operations in $\mathbb{R}$.

References

[GG99] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.

[TCS02] R.L. Rivest T.H. Cormen, C.E. Leiserson and C. Stein. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2002.