Foundations of Computer Algebra: Assignment 1

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

Posted: Friday, January 20, 2006
Due: Monday, February 6, 2006

1 Guidelines

Date of postage. Thursday, January 30, 2006.

Normal date for submission. Thursday, February 3, 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

We consider the following classical algorithm, which computes the gcd of two integer numbers.

$\mathrm{gcd}(a : \mathbb{Z}, b : \mathbb{Z}) : \mathbb{Z} ==$

if $a = b$ **then return** a
if $a \equiv 0 \mod 2$ **and** $b \equiv 0 \mod 2$ **then return** $2\ \mathrm{gcd}(a/2, b/2)$
if $a \equiv 0 \mod 2$ **and** $b \equiv 1 \mod 2$ **then return** $\mathrm{gcd}(a/2, b)$
if $a \equiv 1 \mod 2$ **and** $b \equiv 0 \mod 2$ **then return** $\mathrm{gcd}(a, b/2)$
if $a > b$ **then return** $\mathrm{gcd}((a-b)/2, b)$ **else return** $\mathrm{gcd}((b-a)/2, a)$

Let n_a and n_b the number of bits (in binary-expansion) of a and b respectively.

2.2 Questions

(1) Show that the above algorithm $\mathrm{gcd}(a, b)$ computes the GCD of a and b.

(2) Assume that a and b are two machine integers. Let n_a and n_b the number of bits (in binary-expansion) of a and b respectively. Give an upper bound for the number of machine word operations (comparison of two machine integers, subtraction, shift, accessing a bit of a machine integer) needed for computing $\mathrm{gcd}(a, b)$.

(3) Adapt the above algorithm for computing GDCs in $\mathbb{Z}/2\mathbb{Z}[x]$, that is for univariate polynomials with integer coefficients modulo 2.

2.3 Hints

(1) For the input a and b, we define $c(a, b) = n_a + n_b$. Observe that after in each recursive call, this quantity deceases strictly. So, it is enough to prove the algorithm by induction on $c(a, b)$, which is easy.

(2) Two observations

- there is at most $n_a + n_b$ recursive calls.

   - each recursive call is either terminal or leads to another recursive call plus at most 3 operations in $\mathbb{Z}$: multiplication by 2, division by 2, subtraction. Each of these operations in $\mathbb{Z}$ amounts to a number of word-operations which is linear in the size of the arguments.

   Therefore, the algorithm runs in $O(n_a + n_b)$ operations in $\mathbb{Z}$ and in $O((n_a + n_b)^2)$ word operations.

(3) Algorithm in $\mathbb{Z}/2\mathbb{Z}[x]$:

gcd(a, b) ==
   if $a = b$ **then return** a
   if $a(0) = 0$ **and** $b(0) = 0$ **then return** x gcd$(a/x, b/x)$
   if $a(0) = 0$ **and** $b(0) = 1$ **then return** gcd$(a/x, b)$
   if $a(0) = 1$ **and** $b(0) = 0$ **then return** gcd$(a, b/x)$
   if deg$a >$ deg$b$ **then return** gcd$((a - b)/x, b)$ **else return** gcd$((b - a)/x, a)$

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, 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?

Question (2) is a bit difficult and requires to study carefully the sections of the *Introduction* chapter on complexity estimates.

3.3 Hints

(1) Write $A = A_2X^{2n/3} + A_1x^{n/3} + A_0$ and $B = B_2X^{2n/3} + B_1x^{n/3} + B_0$ where $A_2, A_1, A_0, B_2, B_1, B_1$ are polynomials of degree less than $n/3$. We want to compute $C = AB$. We have

$$C = C_4X^{4n/3} + C_3X^{3n/3} + C_2X^{2n/3} + C_1x^{n/3} + C_0$$

with

$$\begin{array}{rcl} C_4 & = & A_2B_2 \\ C_0 & = & A_0B_0 \\ C_3 & = & (A_2 + A_1)(B_2 + B_1) - A_1B_1 - C_4 \\ C_2 & = & (A_2 + A_0)(B_2 + B_0) - C_0 - C_4 + A_1B_1 \\ C_1 & = & (A_1 + A_0)(B_1 + B_0) - C_0 - A_1B_1 \end{array}$$

Note that each of the polynomials C_0, C_1, C_2, C_3, C_4 has degree less than $2n/3$. Computing the *coefficients* C_0, C_1, C_2, C_3, C_4 requires

- 6 additions in degree less than $n/3$.
- 6 multiplications in degree less than $n/3$.
- 7 additions in degree less than $2n/3$.

Once the *coefficients* C_0, C_1, C_2, C_3, C_4 have been computed the true coefficients of C are not known yet. For intsance, both $C_1x^{n/3}$ and C_0 contain terms in degree d for $n/3 \leq d < 2n/3$. In fact, we need 5 additions in degree less than $n/3$ in order to complete the job.

(2) Let $T(n)$ be the number of operations on the coefficients required to compute C from A and B. We have

$$T(n) = 6T(n/3) + 2n + 14n/3 + 4n/3 = 6T(n/3) + 8n.$$

Using the appropiate formula from [TCS02]. we obtain

$$T(n) \in O(n^{\log_3(6)}).$$

4 Exercise 3

4.1 Statement

Let a and b be two univariate polynomials in $\mathbb{Q}[x]$, 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 $\mathbb{Q}[x]$ 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) 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.

References

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

[KO63] A. Karatsuba and Yu. Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, (7):595–596, 1963.

[Str69] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik.*, 13:354–356, 1969.

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