Bordeaux 1 University
Mathematics

MHT 933 - master 2
2009-2010

Exercises for Chapter 1

Exercise 1 – [MODULAR INVERSION]

1. Write an algorithm to compute $n^k \bmod m$ (n, m, k integers, n, $m > 0$, $k \geq 0$) with the square and multiply method (recursive or iterative version).

2. Let p be a prime number. Using Fermat's little theorem and the previous algorithm, write an algorithm of inversion mod p, i.e. which gives $n^{-1} \bmod p$ if $0 < n < p$.

3. Compute the arithmetic complexity of this algorithm.

Exercise 2 – [FAST FIBONACCI]

We recall that the Fibonacci's sequence is defined by

$$F_0 = 0,\ F_1 = 1,\ \text{et } F_{n+2} = F_{n+1} + F_n \ \text{ for every } n \geq 0.$$

1. Design an algorithm which computes F_n in $O(n)$ additions in $\mathbb{Z}_{\geq 0}$.

2. Give an estimation of the word complexity of this algorithm[1].

3. Show that for every $k, n \in \mathbb{Z}_{\geq 0}$, we have

$$F_{n+k+1} = F_n F_k + F_{n+1} F_{k+1}.$$

4. Deduce from this an algorithm [2] which computes F_n in $O(\log n)$ operations in $\mathbb{Z}_{\geq 0}$.

5. Give an estimation of the word complexity of this new algorithm.

Exercise 3 – [STRASSEN]

Let A and B be two matrices $n \times n$. We want to compute $C = AB$ in an

[1] We recall that $F_n = \frac{1}{\sqrt{5}}\Big(\Phi^n - (-\Phi)^{-n}\Big)$ where $\Phi = (1+\sqrt{5})/2 \approx 1.618$ is the gold number.

[2] You can search for a procedure which computes (F_n, F_{n+1}) from $(F_{n/2}, F_{n/2+1})$ if n is even and $(F_{(n-1)/2}, F_{(n+1)/2})$ if n is odd, procedure which will be used recursively.

economic way. We suppose first that n is a power of 2. We divide A, B and C in 4 matrices $n/2 \times n/2$.

$$A = \begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix}, \quad B = \begin{pmatrix} B_1 & B_2 \\ B_3 & B_4 \end{pmatrix} \quad \text{and} \quad C = \begin{pmatrix} C_1 & C_2 \\ C_3 & C_4 \end{pmatrix}.$$

We put

$$\begin{cases} P_1 &= A_1(B_2 - B_4) \\ P_2 &= (A_1 + A_2)B_4 \\ P_3 &= (A_3 + A_4)B_1 \\ P_4 &= A_4(B_3 - B_1) \\ P_5 &= (A_1 + A_4)(B_1 + B_4) \\ P_6 &= (A_2 - A_4)(B_3 + B_4) \\ P_7 &= (A_1 - A_3)(B_1 + B_2) \end{cases}$$

1. Write C_2 in function of P_1 and P_2, C_3 in function of P_3 and P_4, C_1 in function of $P_4 + P_5$ and $P_2 - P_6$ and C_4 in function of $P_1 + P_5$ and $P_3 + P_7$.

2. Compute the number of additions and of multiplications of matrices $n/2 \times n/2$ which are sufficient to obtain C with this approach.

3. Compute the total number of mutiplications that we are led to make with the algorithm defined by iteration of the process.

4. Find an induction formula for c_n where c_n is the arithmetic complexity of the algorithm for matrices $n \times n$ (total number of operations).

5. Suppose now that n is not necessarily a power of 2. Generalize the previous algorithm to this general case and prove that the algorithm that is obtained has a complexity in

$$O\left(n^{\log 7/\log 2}\right).$$

6. Compare with the complexity of the naive algorithm making use of the formulae

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}.$$

Exercise 4 – [Toom-Cook]

Here we study a generalization of Karatsuba's algorithm. Karatsuba's idea is to reduce the computation of the product of two polynomials of degree(s)

$< 2N$ to the computation of 3 products of polynomials of degree(s) $< N$. But if instead of cutting in two, we cut in three and more generally in k parts, what can we obtain ?

Let us begin with cutting in three parts and suppose that we want to multiply two polynomials P and Q of degree(s) $< n = 3N$. Decompose them in the following way:

$$\begin{cases} P &= P_2X^{2N} + P_1X^N + P_0 \\ Q &= Q_2X^{2N} + Q_1X^N + Q_0 \end{cases}$$

where the P_i and Q_i are polynomials of degree(s) $< N$. Let Π be the product of P and Q and put

$$\begin{cases} \Pi_0 &= P_0Q_0 \\ \Pi_1 &= (P_2 + P_1 + P_0)(Q_2 + Q_1 + Q_0) \\ \Pi_{-1} &= \big(P_2 - P_1 + P_0\big)\big(Q_2 - Q_1 + Q_0\big) \\ \Pi_2 &= \big(4P_2 + 2P_1 + P_0\big)\big(4Q_2 + 2Q_1 + Q_0\big) \\ \Pi_\infty &= P_2Q_2 \end{cases}$$

1. Show that there exists 5 polynomials of degree(s) $< 2N - 1$ denoted by R_i $(0 \le i \le 4)$ such that

$$\begin{cases} \Pi &= \sum\limits_{i=0}^{4} R_iX^{iN} \\ \Pi_\alpha &= \sum\limits_{i=0}^{4} R_i\alpha^i \quad \text{if } \alpha \in \{0, 1, -1, 2\} \\ \Pi_\infty &= R_4 \end{cases}$$

2. Write the R_i $(0 \le i \le 4)$ as functions of the Π_α $(\alpha \in \{0, 1, -1, 2, \infty\})$.

3. Deduce from above questions that the computation of Π can be reduced to the computation of 5 products of 2 polynomials of degree $< N$.

4. Compute the arithmetic complexity (in terms of number of multiplications as a function of n) of the algorithm obtained by using this method (suppose first that n is a power of 3) and compare with Karatsuba.

5. See in what the idea is the same as in Karatsuba (evaluation-interpolation) and generalize.

Exercise 5 – [Mergesort]

We want to sort n numbers in increasing order using only comparisons. We denote by $f(n)$ the number of comparisons required. The main idea is the following one : first, sort the "first half", then sort the "second half" and finally merge. Show that $f(n) = O(n \log n)$.

Exercise 6 – [p-ADIC INVERSION BY NEWTON'S METHOD]

Let p, $l \in \mathbb{Z}_{>0}$ and f, $g_0 \in \mathbb{Z}$ such that $fg_0 \equiv 1 \mod p$. In particular f is inversible modulo p. We want to construct from g_0 an element $g \in \mathbb{Z}$ such that $fg \equiv 1 \mod p^l$. We consider the sequence defined by the induction formula

$$g_{i+1} = 2g_i - fg_i^2 \mod p^{2^{i+1}}.$$

1. Show that for every i we have $fg_i \equiv 1 \mod p^{2^i}$ and that if $r = \lceil \log l / \log 2 \rceil$, the number g_r is an answer to our problem.

2. Analyze in detail the differents steps of the algorithm. Compute its arithmetic and its word complexities.

Exercise 7 – [FAST EUCLIDEAN DIVISION BY NEWTON'S METHOD]

Let S, $T \in A[X]$ where A is a commutative ring with unity 1, and where $\deg(S) = n$, $\deg(T) = m$, $n \geq m$ and T is monic.

1. Show that the classical Euclidean division algorithm of S by T has $O(n^2)$ arithmetic complexity.

2. If $P \in A[X]$ and $k \geq \deg(P)$, we put $Rec_k(P(X)) = X^k P(1/X)$. Show that

$$Rec_{n-m}(Q) = Rec_n(S).Rec_m(T)^{-1} \mod X^{n-m+1},$$

where Q is the quotient in the Euclidean division of S by T.

3. Let $F \in A[X]$ with $F(0) = 1$ and let $l \geq 1$. Let define a sequence of polynomials $G_i \in A[X]$ by $G_0 = 1$ and

$$G_{i+1} = 2G_i - F.G_i^2 \mod X^{2^{i+1}}$$

for $i \geq 0$. Show that for every $i \geq 0$ we have

$$F.G_i \equiv 1 \mod X^{2^i}.$$

4. Deduce from previous questions an algorithm to compute Q and the rest R of the Euclidean division of S by T.

5. Show that the arithmetic complexity of this new algorithm applied to Euclidean division of polynomials of degree $< n$ is in $O(M(n))$ where $M(n)$

is the arithmetic complexity for the computation of the product of 2 polynomials with degree $< n$.

6. Show that we can compute the Euclidean division of 2 integers of length $< n$ with $O(n \log n \log \log n)$ word complexity.

Exercise 8 – [Around FFT]

Let $n = 2^k$ where $k \in \mathbb{Z}_{\geq 0}$. Let R be a commutative ring having as an element a primitive n-th root of unity ω, and in which 2 is invertible (we shall denote its inverse by 1/2). Let P and Q be two polynomials in $R[X]$ with degrees $< n$. We have seen an algorithm (FFT) which computes $P \star Q = PQ \bmod (X^n - 1)$ and thus PQ if $\deg P + \deg Q < n$, which uses the evaluation of P and Q at the ω^i, $0 \leq i \leq n-1$. Here we want to study a variant of this algorithm.

Algorithm 1. Fast Convolution

Require: P, $Q \in R[X]$ with degrees $< n = 2^k$ and the powers $\omega, \omega^2, \ldots, \omega^{n/2-1}$ of a n-th primitive root of unity.

Ensure: $P \star Q = PQ \bmod (X^n - 1)$.

if $k = 0$ **then**

  Return PQ

$P_0 \leftarrow P \bmod (X^{n/2} - 1)$, $P_1 \leftarrow P \bmod (X^{n/2} + 1)$,
$Q_0 \leftarrow Q \bmod (X^{n/2} - 1)$, $Q_1 \leftarrow Q \bmod (X^{n/2} + 1)$.

Call the algorithm **recursively** to compute R_0, $R_1 \in R[X]$ with degrees $< n/2$ such that

$$R_0 \equiv P_0 Q_0 \bmod (X^{n/2} - 1), \; R_1(\omega X) \equiv P_1(\omega X) Q_1(\omega X) \bmod (X^{n/2} - 1).$$

Return

$$\frac{1}{2}\Big((R_0 - R_1)X^{n/2} + R_0 + R_1\Big).$$

---

1. Show that $R_0 \equiv PQ \bmod (X^{n/2} - 1)$ and $R_1 \equiv PQ \bmod (X^{n/2} + 1)$.

2. Deduce from this that the algorithm works correctly.

3. What is its algebraic complexity (number of operations in R)?