Fast Polynomial Multiplication

Marc Moreno Maza

CS 9652, October 4, 2017

Plan

Plan

Zero-divisors

In the whole section we consider a commutative ring R with units.

Definition

An element $a \in R$ is a *zero-divisor* if $a \neq 0$ and there exists $b \in R$ such that $a\,b = 0$ and $b \neq 0$.
If R has no zero-divisors, we say that R is an *integral domain*.

Remark

Observe also the ring R may contain nonzero elements that are neither zero divisor, nor units.
Let $R = \mathbb{Z}/6\mathbb{Z}$ and let $U = R[x]$ be the ring of univariate polynomials over R. The ring U has units (for instance the constant polynomial 5), it has zero divisors (for instance the polynomial $2x + 4$ since $3(2x + 4) = 0$) and also elements like $x + 1$ which is not a unit nor a zero divisor.

Primitive roots of unity (1/2)

Definition

Let n be a positive integer and $\omega \in R$.

1. ω is a *n-th root of unity* if $\omega^n = 1$.
2. ω is a *primitive n-th root of unity* if
   2.1 $\omega^n = 1$,
   2.2 n is a unit in R,
   2.3 for every prime divisor t of n the element $\omega^{n/t} - 1$ is neither zero nor a zero divisor.

Remark

When R is an integral domain the last condition becomes: for every prime divisor t of n the element $\omega^{n/t} \neq 1$. Observe also that a n-th root of unity is necessarily a unit.

Example

Consider that R is the field $\mathbb{C}$ of complex numbers. The number $\omega = e^{2\,\imath\pi/8}$ is a primitive 8-th root of unity.

Primitive roots of unity (2/2)

Example

In $R = \mathbb{Z}/8\mathbb{Z}$ we have $3^2 \equiv 1$. However 3 is not a primitive 2-th of unity, since $n = 2$ is not a unit in R.

Example

In $R = \mathbb{Z}/17\mathbb{Z}$ we have the following computation in AXIOM

```
(1) -> R := PF(17)

   (1)  PrimeField 17
                                             Type: Domain
(2) -> w: R := 3

   (2)  3
                                       Type: PrimeField 17
(3) -> [w^i for i in 0..16]

   (3)  [1,3,9,10,13,5,15,11,16,14,8,7,4,12,2,6,1]
                                    Type: List PrimeField 17
(4) -> u: R := 2

   (4)  2
                                         Type: PrimeField 17
(5) -> [u^i for i in 0..16]

   (5)  [1,2,4,8,16,15,13,9,1,2,4,8,16,15,13,9,1]
                                     Type: List PrimeField 17
```

The first list shows that 3 is a primitive 16-th root of unity. However with $\omega = 2$ we have $\omega^8 - 1 = 0$ since $(2^4 - 1)(2^4 + 1) = 15 \times 17 \equiv 0$.

Properties of primitive roots of unity (1/3)

Proposition

Let $1 < \ell < n$ be integers and let ω be a primitive n-th root of unity. Then we have

1. $\omega^{\ell} - 1$ *is neither zero nor a zero divisor in* R,
2. $\Sigma_{0 \leq j < n}\, \omega^{j\,\ell} = 0$.

Proof (1/3)

It relies on the formula

$$(c - 1)\ \Sigma_{0 \leq j < m}\ c^{j} \quad = \quad c^{m} - 1 \tag{1}$$

which holds for every $c \in R$ and every positive integer m.
Let us prove the first statement of the lemma. Let g be the gcd of ℓ and n. Let $u, v \in \mathbb{Z}$ be such that

$$u\,\ell + v\,n \;=\; g \tag{2}$$

Since $\ell \, < \, n$ we have $1 \, \leq \, g \, < \, n$.

Properties of primitive roots of unity (2/3)

Proof (2/3)

Hence, there exists a prime factor t of n such that

$$g \mid (n/t) \tag{3}$$

Let

$$c = \omega^g \quad \text{and} \quad m = n/(tg) \tag{4}$$

in Relation (1) leading to

$$(\omega^g - 1)\, a = (\omega^{n/t} - 1) \tag{5}$$

for some $a \in R$. Hence if $(\omega^g - 1)$ would be zero or a zero divisor then so would be $(\omega^{n/t} - 1)$ which is false. Now applying Relation (1) with $c = \omega^\ell$ and $m = u$ implies that $(\omega^\ell - 1)$ divides $(\omega^{u\,\ell} - 1)$. But with Relation (2) we obtain

$$(\omega^{u\,\ell} - 1) = (\omega^{u\,\ell}\omega^{v\,n} - 1) = (\omega^g - 1) \tag{6}$$

Hence

$$(\omega^\ell - 1) \mid (\omega^g - 1) \tag{7}$$

Therefore $(\omega^\ell - 1)$ cannot be zero or a zero divisor.

Properties of primitive roots of unity (3/3)

Proof (3/3)

Now let us prove the second statement of the lemma. By applying Relation (1) with $c = \omega^\ell$ and $m = n$ we have

$$(\omega^\ell - 1)\ \Sigma_{0 \leq j < n}\ \omega^{\ell j} \quad = \quad \omega^{\ell\, n} - 1 \quad = 0 \tag{8}$$

Since $(\omega^\ell - 1)$ is neither zero nor a zero divisor we otain the desired formula.

Plan

Discrete Fourier Transform (1/3)

Notations

Let n be a positive integer and $\omega \in R$ be a primitive n-th root of unity. In what follows we identify every univariate polynomial

$$f \; = \; \Sigma_{0 \leq i < n} \, f_i \, x^i \; \in R[x] \tag{9}$$

of degree less than n with its coefficient vector $(f_0, \ldots, f_{n-1}) \in R^n$.

Definition

The R-linear map

$$DFT_\omega \, : \, \left\{ \begin{array}{ccc} R^n & \longmapsto & R^n \\ f & \longmapsto & (f(1), f(\omega), f(\omega^2), \ldots, f(\omega^{n-1})) \end{array} \right. \tag{10}$$

which evaluates a polynomial at the powers of ω is called the *Discrete Fourier Transform* (DFT).

Proposition

The R-linear map DFT_ω is an isomorphism.

Discrete Fourier Transform (2/3)

Proof

- Since the R-linear map DFT_ω is an endomorphism (the source and target spaces are the same) we only need to prove that DFT_ω is bijective.
- Observe that the Vandermonde matrix $VDM(1, \omega, \omega^2, \ldots, \omega^{n-1})$ is the matrix of the R-linear map DFT_ω.
- Then for proving that DFT_ω is bijective we need only to prove that $VDM(1, \omega, \omega^2, \ldots, \omega^{n-1})$ is invertible which holds iff the values $1, \omega, \omega^2, \ldots, \omega^{n-1}$ are pairwise different.
- A relation $\omega^i = \omega^j$ for $0 \leq i < j < n$ would imply $\omega^i \, (1 - \omega^{j-i}) = 0$.
- Since $(1 - \omega^{j-i})$ cannot be zero or a zero divisor then ω^i and thus ω must be zero.
- Then ω cannot be a root of unity. A contradiction.
- Therefore the values $1, \omega, \omega^2, \ldots, \omega^{n-1}$ are pairwise different and DFT_ω is an isomorphism.

Discrete Fourier Transform (3/3

Proposition

Let V_ω denote the matrix of the isomorphism DFT_ω. Then ω^{-1} the inverse of ω is also a primitive n-th root of unity and we have

$$V_\omega \, V_{\omega^{-1}} \;=\; nI_n$$

where I_n denotes the unit matrix of order n.

Proof

Define $\omega' = \omega^{-1}$. Observe that $\omega' = \omega^{n-1}$. Thus ω' is a root of unity, and, in fact, a n-th root of unity. Consider the product of the matrix V_ω and $V_{\omega'}$. The element at row i and column k is:

$$\begin{array}{rcl} (V_\omega \, V_{\omega'})_{ik} & = & \Sigma_{0 \leq j < n} \, (V_\omega)_{ij} (V_{\omega'})_{jk} \\ & = & \Sigma_{0 \leq j < n} \, \omega^{ij} \, (\omega')^{jk} \\ & = & \Sigma_{0 \leq j < n} \, \omega^{ij} \, \omega^{-jk} \\ & = & \Sigma_{0 \leq j < n} \, (\omega^{i-k})^j \end{array}$$

Observe that ω^{i-k} is either a power of ω or a power of its inverse. Thus, in any case this is a power of ω. If $i = k$ this power is 1 and $(V_\omega \, V_{\omega'})_{ik}$ is equal to n. If $i \neq k$, the previous section implies $(V_\omega \, V_{\omega'})_{ik} = 0$.

Plan

n-th convolution of two polynomials (1/2)

Let n be a positive integer and $\omega \in R$ be a primitive n-th root of unity.

Definition

The *convolution* w.r.t. n of the polynomials $f = \Sigma_{0 \leq i < n} f_i x^i$ and $g = \Sigma_{0 \leq j < n} g_j x^j$ in $R[x]$ is the polynomial

$$h = \Sigma_{0 \leq k < n} h_k x^k \tag{11}$$

such that for every $k = 0 \cdots n-1$ the coefficient h_k is given by

$$h_k = \Sigma_{i+j \equiv k \mod n} f_i g_j \tag{12}$$

The polynomial h is denoted by $f *_n g$, or simply by $f * g$ if not ambiguous.

n-th convolution of two polynomials (2/2)

Remark

Observe that the product of f by g is

$$p \;=\; \Sigma_{0 \leq k < 2n-1} \;\; p_k \, x^k \tag{13}$$

where for every $k = 0 \cdots 2n-2$ the coefficient p_k is given by

$$p_k \;=\; \Sigma_{i+j=k} \;\; f_i \, g_j \tag{14}$$

We can rearrange the polynomial p as follows.

$$\begin{array}{rcl} p & = & \Sigma_{0 \leq k < n} \, (p_k \, x^k) \;\; + \;\; x^n \, \Sigma_{0 \leq k < n-1} \, (p_{k+n} \, x^k) \\ & = & \Sigma_{0 \leq k < n} \, (p_k \; + \; p_{k+n} \, x^n) \, x^k \\ & \equiv & \Sigma_{0 \leq k < n} \, h_k \, x^k \quad \bmod \; x^n - 1 \end{array} \tag{15}$$

with $p_{2n-1} = 0$, since $\deg(p) = \deg(f) + \deg(g) = 2n-2$.
Therefore we have

$$f * g \;\; \equiv \;\; fg \quad \bmod \; x^n - 1 \tag{16}$$

DFT and convolution

Proposition

For $f, g \in R[x]$ univariate polynomials of degree less than n we have

$$DFT_\omega(f * g) = DFT_\omega(f) DFT_\omega(g) \tag{17}$$

where the product of the vectors $DFT_\omega(f)$ and $DFT_\omega(g)$ is computed component-wise.

Proof

Since $f * g$ and $f\,g$ are equivalent modulo $x^n - 1$, there exists a polynomial $q \in R[x]$ such that

$$f * g = f\,g + q\,(x^n - 1) \tag{18}$$

Hence for $i = 0 \cdots n - 1$ we have

$$\begin{aligned} (f * g)(\omega^i) &= f(\omega^i)\,g(\omega^i) + q(\omega^i)(\omega^{i\,n} - 1) \\ &= f(\omega^i)\,g(\omega^i) \end{aligned} \tag{19}$$

since $\omega^n = 1$.

Plan

A divide-and-conquer strategy (1/4)

The Fast Fourier Transform computes the DFT quickly. This important algorithm for computer science (not only computer algebra, but also digital signal processing for instance) was (re)-discovered in 1965 by Cooley and Tukey.

- Let n be a positive **even** integer, $\omega \in R$ be a primitive n-th root of unity and $f = \sum_{0 \leq i < n} f_i x^i$.
- In order to evaluate f at $1, \omega, \omega^2, \ldots, \omega^{n-1}$, we follow a *divide-and-conquer* strategy; more precisely, we consider the divisions with remainder of f by $x^{n/2} - 1$ and $x^{n/2} + 1$.
- So let q_0, q_1, r_0, r_1 be polynomials such that

$$f = q_0(x^{n/2} - 1) + r_0 \quad \text{with} \quad \begin{cases} \deg(r_0) < n/2 \\ \deg(q_0) < n/2 \end{cases} \tag{20}$$

and

$$f = q_1(x^{n/2} + 1) + r_1 \quad \text{with} \quad \begin{cases} \deg(r_1) < n/2 \\ \deg(q_1) < n/2 \end{cases} \tag{21}$$

A divide-and-conquer strategy (2/4)

- The relations $\deg(q_0) < n/2$ and $\deg(q_1) < n/2$ hold because the polynomial f has degree less than n.
- Observe that the computation of (q_0, r_0) and (q_1, r_1) can be done very easily.
- Indeed, let $F_0, F_1 \in R[x]$ be such that

$$f = F_1 x^{n/2} + F_0 \quad \text{with} \quad \begin{cases} \deg(F_1) < n/2 \\ \deg(F_0) < n/2 \end{cases} \tag{22}$$

  We have

$$f = F_1(x^{n/2} - 1) + F_0 + F_1 \quad \text{and} \quad f = F_1(x^{n/2} + 1) + F_0 - F_1 \tag{23}$$

- Hence we obtain

$$r_0 = F_0 + F_1 \quad \text{and} \quad r_1 = F_0 - F_1 \tag{24}$$

A divide-and-conquer strategy (3/4)

- Let i be an integer such that $0 \leq i < n/2$. By using Relation (20) with $x = \omega^{2\,i}$ we obtain

$$f(\omega^{2\,i}) \;=\; q_0(\omega^{2\,i})(\omega^{n\,i} - 1) + r_0(\omega^{2\,i}) \;=\; r_0(\omega^{2\,i}) \tag{25}$$

  since $\omega^{n\,i} = 1$.

- Then, by using Relation (21) with $x = \omega^{2\,i+1}$ we obtain

$$f(\omega^{2\,i+1}) \;=\; q_1(\omega^{2\,i+1})(\omega^{n\,i}\,\omega^{n/2} + 1) + r_1(\omega^{2\,i+1}) \;=\; r_1(\omega^{2\,i+1}) \tag{26}$$

  since $\omega^{n/2} = -1$.

- Indeed, this last equation follows from

$$0 \;=\; \omega^{n} - 1 \;=\; (\omega^{n/2} - 1)(\omega^{n/2} + 1) \tag{27}$$

  and the fact that $\omega^{n/2} - 1$ is not zero nor a zero divisor.

A divide-and-conquer strategy (4/4)

Therefore we have proved the following.

Proposition

Evaluating $f \in R[x]$ *(with degree less than* n*) at* $1, \omega^1, \ldots, \omega^{n-1}$ *is equivalent to*

- *evaluate* r_0 *at the even powers* ω^{2i} *for* $0 \leq i < n/2$*, and*
- *evaluate* r_1 *at the odd powers* ω^{2i+1} *for* $0 \leq i < n/2$*.*

- Since it is easy to show that ω^2 is a primitive $n/2$-th root of unity we can hope for a recursive algorithm.
- This algorithm would be easier if both r_0 and r_1 would be evaluated at the same points. So we define

$$r_1{}^*(\omega^{2i}) = r_1(\omega^{2i+1}). \tag{28}$$

Algorithm

Input: $n = 2^k$, $f = \Sigma_{0 \le i < n} f_i x^i$, and the powers $1, \omega, \omega^2, \ldots, \omega^{n-1}$ of a primitive n-th root of unity $\omega \in R$.

Output: $DTF_\omega(f) = (f(1), f(\omega), f(\omega^2), \ldots, f(\omega^{n-1}))$.

if $n = 1$ **return** (f_0)
$r_0 := \Sigma_{0 \le j < n/2} (f_j + f_{j+n/2}) x^j$
$r_1^* := \Sigma_{0 \le j < n/2} \omega^j (f_j - f_{j+n/2}) x^j$
call the algorithm recursively to evaluate r_0 and r_1^*
at the $n/2$ first powers of ω^2
return$(r_0(1), r_1^*(1), r_0(\omega^2), r_1^*(\omega^2), \ldots, r_0(\omega^{n-2}), r_1^*(\omega^{n-2}))$

Complexity analysis

Proposition

Let n be a power of 2 and $\omega \in R$ be a primitive n-th root of unity. Then, the previous FFT algorithm computes $DTF_\omega(f)$ using

- $n \log(n)$ *additions in* R,
- $(n/2) \log(n)$ *multiplications by powers of* ω.

leading in total to $3/2\, n \log(n)$ *ring operations.*

Proof

By induction on $k = \log_2(n)$. Let $S(n)$ and $T(n)$ be the number of additions and multiplications in R that the algorithms requires for an input of size n. If $k = 0$ the algorithm returns (f_0) whose costs is null thus we have $S(0) = 0$ and $T(0) = 0$ which satisfies the formula since $\log(n) = \log(1) = 0$. Assume $k > 0$. Just by looking at the algorithm we that

$$S(n) \;=\; 2\,S(n/2) + n \quad \text{and} \quad T(n) \;=\; 2\,T(n/2) + n/2 \qquad (29)$$

leading to the result by plugging in the induction hypothesis.

Plan

Algorithm

Input: $f, g \in R[x]$ with degree less than $n = 2^k$, a primitive n-th root of unity $\omega \in R$.

Output: $f * g \in R[x]$

compute $1, \omega, \omega^2, \ldots, \omega^{n-1}$
$\alpha := DFT_\omega(f)$
$\beta := DFT_\omega(g)$
$\gamma := \alpha\,\beta$
return $(DFT_\omega)^{-1}(\gamma) = 1/n\, DFT_{\omega^{-1}}(\gamma)$

Complexity analysis

Proposition

Let n be a power of 2 and $\omega \in R$ a primitive n-th root of unity. Then convolution in $R[x]/\langle x^n - 1\rangle$ and multiplication in $R[x]$ of polynomials whose product has degree less than n can be performed using

- $3\, n \log(n)$ *additions in* R,
- $3/2\, n \log(n) + n - 2$ *multiplications by a power of* ω,
- n *multiplications in* R,
- n *divisions by* n *(as an element of* R*)*,

leading to $9/2\, n \log(n) + \mathcal{O}(n)$ *operations in* R.

Remark

To multiply two arbitrary polynomials of degree less than $n \in \mathbb{N}$ we only need a primitive 2^k-th root of unity where

$$2^{k-1} < 2\, n \leq 2^k \tag{30}$$

Then we have decreased the cost of about $\mathcal{O}(n^2)$ of the classical algorithm to $\mathcal{O}(n \log(n))$.

The multivariate case is discussed here: `http://www.csd.uwo.ca/~moreno/CS433-CS9624/BPAS-CS9624-2x2.pdf`

Plan

Computing primitive roots

Please read this section: http://www.csd.uwo.ca/~moreno//CS424/Lectures/FastMultiplication.html/node7.html
In particular, the subsection: http://www.csd.uwo.ca/~moreno//CS424/Lectures/FastMultiplication.html/node10.html

Plan

Efficient implementation of FFT

Please read this section: `http://www.csd.uwo.ca/~moreno//CS424/Lectures/FastMultiplication.html/node11.html`

Plan

Algorithm

Input: univariate polynomials $a = \Sigma_0^n\ a_i x^i$ and $b = \Sigma_0^m\ b_i x^i$ in $R[x]$ with respective degrees n and m such that $n \geq m \geq 0$ and b_m is a unit.

Output: the quotient q and the remainder r of a w.r.t. b. Hence $a = bq + r$ and $\deg r < m$.

$r := a$
for $i = n - m, n - m - 1, \ldots, 0$ **repeat**
    if $\deg r = m + i$ **then**
        $q_i :=$ leadingCoefficient$(r)\ /\ b_m$
        $r := r - q_i x^i b$
    else $q_i := 0$
$q := \Sigma_0^{n-m}\ q_i x^i$
return (q, r)

Complexity analysis (1/2)

Proposition

Let a and b two univariate polynomials in $R[x]$ with respective degrees n and m such that $n \geq m \geq 0$ and the leading coefficient of b is a unit. Then, there exist unique polynomials q and r such that $a = bq + r$ and $\deg r < m$*. The polynomials q and r are called the* quotient *and the* remainder *of a w.r.t. to b. Moreover, the previous algorithm compute them in $(2m+1)(n-m+1) \in \mathcal{O}(n\,m)$ operations in R.*

Proof

- Consider an iteration of the **for** loop where $\deg r = m + i$ holds at the begining of the loop.
- Observe that r and $q_i x^i b$ have the same leading coefficient. Since $\deg b = m$, computing the reductum of $q_i x^i b$ requires m operations.
- Then subtracting the reductum of $q_i x^i b$ to that of r requires again m operations.
- Hence each iteration of the **for** loop requires at most $2m+1$ operations in R, since we need also to count 1 for the computation of q_i.
- The number of **for** loops is $n-m+1$. Therefore, the algorithm requires $(2m+1)(n-m+1)$.

Complexity analysis (2/2)

Remark

- One can derive from the previous algorithm a bound for the coefficients of q and r, which is needed for performing a modular version (based for instance on the CRT).
- Let $||a||$, $||b||$ and $||r||$ be the max-norm of a, b and r. Let $\mid b_m \mid$ be the absolute value (over $\mathbb{Z}$) or the norm (over $\mathbb{C}$) of the leading coefficient of b.
- Then we have

$$||r|| \;\leq\; ||a|| \left(1 + \frac{||b||}{\mid b_m \mid}\right)^{n-m+1} \tag{31}$$

Plan

Prelimianries

- We shall see now that the previous complexity result can be improved.
- To do so, we will show that q can be computed from a and b by performing essentially one multiplication in $R[x]$.
- We start with the equation

$$a(x) \;=\; q(x)\,b(x) + r(x) \tag{32}$$

  where a, b, q and r are in the statement of the previous proposition.
- Replacing x by $1/x$ and multiplying the equation by x^n leads to the new equation:

$$x^n\,a(1/x) \;=\; (x^{n-m}q(1/x))\;(x^m\,b(1/x)) \;+\; x^{n-m+1}\left(x^{m-1}\,r(1/x)\right) \tag{33}$$

- In Equation (33) each of the rational fractions $a(1/x)$, $b(1/x)$, $q(1/x)$ and $r(1/x)$ is multiplied by x^e such that e is an upper bound for the degree of its denominator.
- So Equation (33) is in fact an equation in $R[x]$.

Using reversals (1/2)

Definition

For a univariate polynomial $p = \Sigma_0^d \ p_i x^i$ in $R[x]$ with degree d and an integer $k \geq d$, the *reversal of order* k of p is the polynomial denoted by $\mathrm{rev}_k(p)$ and defined by

$$\mathrm{rev}_k(p) \ = \ x^k \, p(1/x) \ = \ \Sigma_{k-d}^k \ p_{k-i} x^i. \quad (34)$$

When $k = d$ the polynomial $\mathrm{rev}_k(p)$ is simply denoted by $\mathrm{rev}(p)$. Hence we have

$$\mathrm{rev}(p) \ = \ p_d + p_{d-1}x + \cdots + p_1 x^{d-1} + p_0 x^d. \quad (35)$$

Proposition

With a, b, q *and* r *as abovce, we have*

$$\mathrm{rev}_n(a) \ \equiv \ \mathrm{rev}_{n-m}(q) \ \mathrm{rev}_m(b) \quad \bmod \ x^{n-m+1} \quad (36)$$

Using reversals (2/2)

Proof

Indeed with the above definition, Equation (33) reads

$$\mathrm{rev}_n(a) \; = \; \mathrm{rev}_{n-m}(q)\ \mathrm{rev}_m(b) + x^{n-m+1}\ \mathrm{rev}_{m-1}(r) \qquad (37)$$

leading to the desised result.

Remark

- If R is a field then we know that $\mathrm{rev}_{n-m}(q)$ is invertible modulo x^{n-m+1}.
- Indeed, $\mathrm{rev}_{n-m}(q)$ has constant coefficient 1 and thus the gcd of $\mathrm{rev}_{n-m}(q)$ and x^{n-m+1} is 1.
- The case where R is not a field leads also to a simple and surprising solution as we shall see in the next section.

Plan

Objective

- Let R be a commutative ring with identity element.
- Given $f \in R[x]$ and $\ell \in \mathbb{N}$ such that $f(0) = 1$ compute the polynomials $g \in R[x]$ such that

$$f\,g \equiv 1 \mod x^{\ell} \quad \text{and} \quad \deg(g) < \ell. \tag{38}$$

First, we observe that if there is a solution, then it is unique.

Unicity result

Proposition

If Equation (38) has a solution $g \in R[x]$ with degree less than ℓ then it is unique.

Proof

- Indeed, let g_1 and g_2 be solutions of Equation (38).
- Then the product $f(g_1 - g_2)$ is a multiple of x^ℓ. Since $f(0) = 1$ then $g_1(0) - g_2(0)$ must be 0.
- Hence there is a constant $c \in R$ and polynomials h_1, h_2 with degree less than $\ell - 1$ such that

$$g_1(x) = h_1(x)\,x + c \quad \text{and} \quad g_2(x) = h_2(x)\,x + c \tag{39}$$

- It follows that $f(h_1 - h_2)$ is a multiple of $x^{\ell-1}$. By repeating the same argument we show that $h_1(0) = h_2(0)$.
- Then by induction on ℓ we obtain $g_1 = g_2$.

Relation to Newton iteration

- Since Equation (38) is an equation in $R[x]/\langle x^\ell \rangle$, a solution of this equation can be viewed as an approximation of a more general problem.
- Think of truncated Taylor expansions!
- So let us recall from numerical analysis the celebrated Newton iteration and let $\phi(g) = 0$ be an equation that we want to solve, where $\phi : \mathbb{R} \longmapsto \mathbb{R}$ is a differentiable function.
- From a suitable initial approximation g_0, the sequence, called *Newton iteration step*,

$$g_{i+1} \; = \; g_i - \frac{\phi(g_i)}{\phi'(g_i)} \tag{40}$$

  allows to compute subsequent approximations and converge toward a desired solution.
- In our case we have $\phi(g) = 1/g - f$ and the Newton iteration step is

$$g_{i+1} \; = \; g_i - \frac{1/g_i - f}{-1/{g_i}^2} \; = \; 2g_i - f\,{g_i}^2. \tag{41}$$

Existence result (1/2)

Proposition

Let R be a commutative ring with identity element. Let f be a polynomial in $R[x]$ such that $f(0) = 1$. Let $g_0, g_1, g_2, \ldots$ be the sequence of polynomials defined for all $i \geq 0$ by

$$\left\{ \begin{array}{rcl} g_0 & = & 1 \\ g_{i+1} & \equiv & 2g_i - f\,{g_i}^2 \mod x^{2^{i+1}}. \end{array} \right. \tag{42}$$

Then for $i \geq 0$ we have

$$f\,g_i \equiv 1 \mod x^{2^i}. \tag{43}$$

Existence result (2/2)

Proof

By induction on $i \geq 0$. For $i = 0$ we have $x^{2^i} = x$ and thus

$$f\, g_i \;\equiv\; f(0)\, g_0 \;\equiv\; 1 \times 1 \;\equiv\; 1 \quad \mathrm{mod} \;\; x^{2^i}. \tag{44}$$

For the induction step we have

$$\begin{array}{rcll} 1 - f\, g_{i+1} & \equiv & 1 - f\,(2 g_i - f\, {g_i}^2) & \mathrm{mod} \;\; x^{2^{i+1}} \\ & \equiv & 1 - 2\, f\, g_i + f^2\, {g_i}^2 & \mathrm{mod} \;\; x^{2^{i+1}} \\ & \equiv & (1 - f\, g_i)^2 & \mathrm{mod} \;\; x^{2^{i+1}} \\ & \equiv & 0 & \mathrm{mod} \;\; x^{2^{i+1}}. \end{array} \tag{45}$$

Indeed $f\, g_i \equiv 1 \mod x^{2^i}$ means that x^{2^i} divides $1 - f\, g_i$. Thus $x^{2^{i+1}} = x^{2^i + 2^i} = x^{2^i}\, x^{2^i}$ divides $(1 - f\, g_i)^2$.

Algorithm

Input: $f \in R[x]$ such that $f(0) = 1$ and $\ell \in \mathbb{N}$.

Output: $g \in R[x]$ such that $f\,g \equiv 1 \mod x^{\ell}$

$g_0 := 1$
$r := \lceil \log_2(\ell) \rceil$
for $i = 1 \cdots r$ **repeat**
    $g_i := \left(2g_{i-1} - f\,{g_{i-1}}^2\right) \mod x^{2^i}$
return g_r

Multiplication time (1/2)

Definition

- A *multiplication time* is a function $\mathbb{M} : \mathbb{N} \longrightarrow \mathbb{R}$ such that for any commutative ring R with a 1, for every $n \in \mathbb{N}$, any pair of polynomials in $R[x]$ of degree less than n can be multiplied in at most $\mathbb{M}(n)$ operations of R.
- In addition, $\mathbb{M}$ must satisfy $\mathbb{M}(n)/n \geq \mathbb{M}(m)/m$, for every $m, n \in \mathbb{N}$, with $n \geq m$.
- This implies the *super-linearity* properties, that is, for every $m, n \in \mathbb{N}$

$$\mathbb{M}(nm) \geq m\mathbb{M}(n), \quad \mathbb{M}(n+m) \geq \mathbb{M}(m) + \mathbb{M}(n) \quad \text{and} \quad \mathbb{M}(n) \geq n. \quad (46)$$

Multiplication time (2/2)

Examples

- Classical: $d \longmapsto 2d^2$;
- Karatsuba: $d \longmapsto C\, d^{\log_2(3)}$ with some C that can be taken equal to 9;
- FFT over an arbitrary ring: $d \longmapsto C\, d \log(d) \log(\log(d))$ for some C that can taken equal to 64.

Note that the FFT-based multiplication in degree d over a ring that supports the FFT (that is, possessing primitive n-th root of unity, where n is a power of 2 greater than $2d$) can run in $C\, d \log(d)$ operations in R, with some $C \geq 18$.

To learn about Karatsuba's multiplication algorithm:
`https://en.wikipedia.org/wiki/Karatsuba_algorithm`
To learn about multiplication times and the complexity of algebraic operations `https://en.wikipedia.org/wiki/Computational_complexity_of_mathematical_operations`

Complexity analysis (1/4)

Proposition

The above algorithm computes the inverse of f modulo x^{ℓ} in $3\mathbb{M}(\ell) + 0(\ell)$ *operations in R.*

Proof (1/3)

- Since x^{ℓ} divides x^{2^r}, the result is also valid modulo x^{ℓ}.
- Before proving the complexity result, we point out the following relation for $i = 1 \cdots r$.

$$g_i \equiv g_{i-1} \mod x^{2^{i-1}} \qquad (47)$$

- Indeed, we have

$$\begin{array}{rcll} g_i & \equiv & 2g_{i-1} - f\, {g_{i-1}}^2 & \mod x^{2^i} \\ & \equiv & 2g_{i-1} - f\, {g_{i-1}}^2 & \mod x^{2^{i-1}} \\ & \equiv & g_{i-1}(2 - f\, g_{i-1}) & \mod x^{2^{i-1}} \\ & \equiv & g_{i-1}(2 - 1) & \mod x^{2^{i-1}} \\ & \equiv & g_{i-1} & \mod x^{2^{i-1}} \end{array} \qquad (48)$$

Complexity analysis (2/4)

Proof (2/3)

Therefore when computing g_i we only care about powers of x in the range $x^{2^{i-1}} \cdots x^{2^i}$. This says that

- half of the computation of g_r is made during the last iteration of the **for** loop,
- a quarter is made when computing g_{r-1} etc.

Now recall that

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots = 1 \tag{49}$$

So **roughly** the cost of the algorithm is in the order of magnitude of the cost of the last iteration. which consists of

- two multiplications of polynomials with degree less than 2^r,
- a multiplication of a polynomial (with degree less than 2^r) by a constant,
- truncations modulo x^{2^r}
- a subtraction of polynomials with degree less than 2^r.

leading to $2\mathbb{M}(2^r) + O(2^r)$ operations in R.

Complexity analysis (3/4)

Proof (3/3)

But this was not a formal proof, although the principle was correct. Let us give a more formal proof.
The cost for the i-th iteration is

- $\mathbb{M}(2^{i-1})$ for the computation of ${g_{i-1}}^2$,
- $\mathbb{M}(2^i)$ for the product $f\, {g_{i-1}}^2 \mod x^{2^i}$,
- and then the opposite of the upper half of $f {g_{i-1}}^2$ modulo x^{2^i} (which is the upper half g_i) takes 2^{i-1} operations.

Thus we have $\mathbb{M}(2^i) + \mathbb{M}(2^{i-1}) + 2^{i-1} \leq \frac{3}{2}\,\mathbb{M}(2^i) + 2^{i-1}$, resulting in a total running time:

$$\sum_{1 \leq i \leq r} \frac{3}{2}\,\mathbb{M}(2^i) + 2^{i-1} \leq \left(\frac{3}{2}\,\mathbb{M}(2^r) + 2^{r-1}\right) \sum_{1 \leq i \leq r} 2^{i-r} < 3\,\mathbb{M}(2^r) + 2^r = 3\,\mathbb{M}(\ell) + \ell \qquad (50)$$

since $\mathbb{M}(n) \leq \frac{1}{2}\mathbb{M}(2n)$ for all $n \in \mathbb{N}$

Complexity analysis (4/4)

Remark

Once again for $i = 1 \cdots r$ we have

$$g_i \equiv g_{i-1} \mod x^{2^{i-1}} \quad (51)$$

So when implementing the above algorithm, one should be extremely careful in not recomputing the *low terms* of g_i that come from g_{i-1}.

Remark

- The above can be adapted to the case where $f(0)$ is a unit different from 1 by initializing g_0 to the inverse of $f(0)$ instead of 1.
- If $f(0)$ is not a unit, then no inverse of f modulo x^ℓ exists.
- Indeed $f\,g \equiv 1 \mod x^\ell$ implies $f(0)g(0) = 1$ which says that $f(0)$ is a unit.

Learn about the Middle Product Technique at: `http://www.csd.uwo.ca/~moreno//CS424/Lectures/FastDivisionAndGcd.html/node4.html` (see the last remark in that section)

Plan

Algorithm

Input: $a, b \in R[x]$ with $b \neq 0$ monic.

Output: q, r such that $a = b\,q + r$ and $\deg r < \deg b$.

$n := \deg a$
$m := \deg b$
if $n < m$ **then**
  $q := 0$
  $r := a$
else
  $f := \mathrm{rev}_m(b)$
  $g :=$ inverse of f modulo x^{n-m+1}
  $q := \mathrm{rev}_n(a)\,g \mod x^{n-m+1}$
  $q := \mathrm{rev}_{n-m}(q)$
  $r := a - b\,q$
return (q, r)

Complexity analysis (1/2)

Proposition

- *Let R be a commutative ring with identity element. Let a and b be univariate polynomials over R with respective degrees n and m such that $n \geq m \geq 0$ and $b \neq 0$ monic.*
- *The above algorithm computes the quotient and the remainder of a w.r.t. b in $4\,\mathbf{M}(n-m) + \mathbf{M}(\max(n-m, m)) + \mathcal{O}(n)$ operations in R*

Proof

Indeed, this algorithm consists essentially in

- 3 multiplications in degree $n-m+1$, plus operations in $O(n-m+1)$ operations in R, in order to compute g (by virtue of the previous section),
- one multiplication in degree $n-m+1$ to compute q,
- one multiplication in degree $\max(n-m, n)$, and one subtraction in degree n to compute r.

Complexity analysis (2/2)

Remark

- If several divisions by a given b needs to be performed then we may precompute the inverse of $\mathrm{rev}_m(b)$ modulo some powers $x, x^2, \ldots$ of x.
- Assuming that R possesses suitable primitive roots of unity, we can also save their DFT.

Remark

- In the above result, the complexity estimate becomes $3\,\mathbf{M}(n-m) + \mathbf{M}(\max(n-m, n)) + \mathcal{O}(n)$ if the *middle product* technique applies.
- Moreover, if $n - m \leq n$ holds, the $\mathcal{O}(n)$ can be replaced by $\mathcal{O}(m)$.

Plan

Fast EEA

Please read this section: http://www.csd.uwo.ca/~moreno//CS424/Lectures/FastDivisionAndGcd.html/node6.html