Sorbonne University

Fondement de l'algorithmique algébrique

MU4IN902

Instructors: Prof. Vincent Neiger

Hayato Ishida

Ver.1.0

Contents

1 Introduction

1.1 Ring and Field

Rough definition : Field is a set F with two operations $(+, \times)$ with "usual" properties (e.g. $a(b+c) = ab+ac$, $a(bc) = (ab)c$ etc...). And every element has an inverse. If one of the elements does not have an inverse, it is a ring.

- So, elements in the field can be inverted except 0.

- Every element $a \in F$ has an opposite for $+$, these act on b s.t. $a + b = 0$. This is true for all rings and fields.

- Therefore, invertibility is about "$\times$"

Remark 1.1

Ring $\longleftarrow$ Field

Examples

- $\mathbb{R}$ the set of real numbers (field). $\mathbb{Q}$ rational (field). $\mathbb{C}$ complex (field).

- $\mathbb{Z}$ integers is not a field, thus a ring. Why?

This is because 2 is not invertible. For instance, there is no $x \in \mathbb{Z}$ s.t. $2x = 1$.

- $\mathbb{Z}/p\mathbb{Z}$ is a field. In general, $a \in \{1, ..., p-1\}$, there are u, v s.t. $au + pv = 1$, so $au = 1 \; mod \; p$.

- $\mathbb{Z}/4\mathbb{Z}$ is not a field, why? $\longrightarrow$ because 2 is not invertible.

1.2 Complexity

In this course, we manipulate algebraic objects. For instance, polynomials (ring), matrices (field), and more. Complexity measures the efficiency of algorithms. Algebraic complexity is counting the number of operations in the base ring and field.

The complexity here does not take into account of ...

- all memory-related operations

- the size of the elements in the base ring or field. (e.g. : addition of polynomials of degree $\leq d$ in $\mathbb{Q}[x]$, algebraic complexity is $O(d)$ and ignore the size of the rational coefficients.

*You should be familiar with the complexity from MODEL.

1.3 Matrices

For a field $\mathbb{K}$ or denote by $\mathbb{K}^{m \times n}$ the set of $m \times n$ matrices over $\mathbb{K}$.

Recall : Not a ring because .... ??

It is represented as a two-dimensional array with rows and columns. $A = (a_{i\,j}) \in \mathbb{K}^{m \times n}$ where $0 \leq i \leq m$ and $0 \leq j \leq n$.

Addition

Complexity is $O(m\,n)$, operations in $\mathbb{K}$.

Multiplication

Multiplication of $A \in \mathbb{K}^{m\times n}$ by $B \in \mathbb{K}^{n\times p}$

- Naive algorithm : $O(m\,n\,p)$

- Strassen's algorithm (recall form MODEL) has complexity $O(n^{log_2(7)})$ where $m = n = p$

- As an alternative, there is a lot of work on matrix multiplication algorithms. The best performance today is $O(n^{2.37})$ but for the moment it is impractical.

1.4 Polynomials

For a field $\mathbb{K}$ or a ring $\mathbb{R}$, the sets $\mathbb{K}[x]$, $\mathbb{R}[x]$ are those of polynomials in one variable x.

$\mathbb{K}[x] = \{P = P_0 + P_1x + P_2x^2 + ... + P_dx^d\}$ in $\mathbb{K}$. (same for $\mathbb{R}$)

They are represented as a one-dimensional array $[P_0, P_1, P_2, ..., P_d]$.

Addition

of two polynomials in $\mathbb{R}[x]$ of degree $\leq d$ has a complexity $O(d)$ operations in $\mathbb{R}$.

Multiplication :

- Naive algorithm : $O(d^2)$ operations in $\mathbb{R}$ - Karatsuba (recall from MODEL) : $O(d^{log_2(3)}) \approx O(d^{1.584})$ operations in $\mathbb{R}$

- For alternative FFT (again, recall from MODEL) : $O(d\ log(d)\ log\ log(d))$

2 Euclid's division and GCD

- $A|B$ means A divides B. Meaning there is no reminder (and A and B are polynomials here)

2.1 Algebraic structure

Definition 2.1

$\mathbb{K}$ field

$f \in \mathbb{K}[x] \Rightarrow f = a_0 + a_1 x + ... + a_n x^n$ with $a_i \in \mathbb{K}$

$n = deg(F)$

- Roots : $\alpha \in \mathbb{K} s.t f(\alpha) = 0$

Lemma 2.0

α is a root of $f \iff (x - \alpha)|f$, in other word $f \in \mathbb{K}[x]$ and $\alpha \in \mathbb{K} s.t\ f(\alpha) = 0$, the $(X - a)|f$

Lemma 2.0

f has degree n, then f has at most n roots.

Definition 2.1: Algebraic closure

- $\overline{\mathbb{K}}$ is the algebraic closure of K if :

- $\mathbb{K} \subset \overline{\mathbb{K}}$

- $\forall f \in \bar{\mathbb{k}}\ [x], f$ has exactly $degf$ roots in $\bar{\mathbb{k}}$.

Example :

$\mathbb{C}$ is algebraic closed.

$\mathbb{R} \subset \mathbb{C}$ and $\mathbb{Q} \subset \mathbb{C}$ are algebraic closure.

But, $\mathbb{R}$ is not $\longrightarrow x^2 + 1$ has 0 roots in $\mathbb{R}$.

Definition 2.2: Recalling a definition of ring

Let R a set equipped with two binary operation s.t $(R, +, \times)$. It is said to be a ring if the followings hold $\longrightarrow$

- $R, +)$ commutative group with neutral 0_R (what is 0_R means ???) - $\times$ has a neutral element 1_R and is associative

$a \times b \times c = (a \times b) \times c = a \times (b \times c)$

- $\times$ is distributive with respect to $+$.

$a(b + c) = ab + ac$

Example :

$\mathbb{Z}, \mathbb{K}[x], \mathbb{R}, \mathbb{C}$, but $\mathbb{N}$ is not a ring because $-1 \in \mathbb{N}$.

Proposition 2.1

$f = a_0 + ... + a_n x^n$ with $a_i \in \mathbb{R}$, $\mathbb{R}[x]$

if R is a ring so is $R[x]$

Definition 2.3: Monic Polynomial

Monic polynomial is a polynomial with a leading coefficient equal to 1.

Example

- Monic polynomial : $x + 1$, $x^2 - 7x + 99$, $x^{1000} + x^{99} - 10000$
- Polynomial that are not monic : $5x^{99} + 4$, $8x^3 + x^2 - 7x + 0$

Definition 2.4: Quotient ring

Let R be a ring and $f \in R[x]$ be monic. Also, $s, t \in R[x]$.

- We say that s and t are equivalent modulo $f \iff s \equiv t \ \ mod \ f$, if f divides $s - t$.
- The set of classes of equivalences is denoted by $R[x]/\langle f \rangle$ and is called quotient ring.

Example :

$x^2 + x + 1 \equiv 1 \ \ mod \ \ x + 1$, since $x^2 + x + 1 = x(x + 1) + 1$

2.2 Division algorithm for polynomials in $\mathbb{K}[x]$

Definition 2.5: Euclidean domain

$\mathbb{R}$ is a Euclidean domain if there exists a Euclidean division (and R is an integral domain).

Algorithm 1: PolynomialDivisionAlgorithm

Input: Two polynomials $A = a_m X^m + \cdots + a_0$ and $B = b_n X^n + \cdot + b_0$ in $\mathbb{K}[X]$.
Output: Two polynomials $Q = q_{m-n} X^{m-n} + \cdots + q_0$ and $R = r_p X^p + \cdots + r_0$ in $\mathbb{K}[X]$ such that $A = BQ + R$ and $p = \deg R < \deg B = n$.

$R := A$, $Q = 0$, $b = \mathrm{lc}(B)$
While $\deg R \geq \deg B$ **do**
  $a := \mathrm{lc}(R)$ it is the leading coefficient of $f = a_0 + \ldots + \underline{a_n} x^n$
  $Q := Q + \frac{a}{b} X^{\deg R - \deg B}$
  $R := R - \frac{a}{b} X^{\deg R - \deg B} B$
Return (Q, R)

Proposition 2.2

On input A and B in $\mathbb{K}[x]$ with degree m and n, with $m > n$. Polynomial division algorithm perform $O(n(m-n))$ arithmetic operation in $\mathbb{K}$.

Algorithm 1: PolynomialDivisionAlgorithm

Input: Two polynomials $A = a_m X^m + \cdots + a_0$ and $B = b_n X^n + \cdot + b_0$ in $\mathbb{K}[X]$.
Output: Two polynomials $Q = q_{m-n} X^{m-n} + \cdots + q_0$ and $R = r_p X^p + \cdots + r_0$ in $\mathbb{K}[X]$ such that $A = BQ + R$ and $p = \deg R < \deg B = n$.

$R := A$, $Q = 0$, $b = \mathrm{lc}(B)$
While $\deg R \geq \deg B$ **do** $m-n$ iterations
  $a := \mathrm{lc}(R)$
  $Q := Q + \frac{a}{b} X^{\deg R - \deg B}$ n operations at each iteration.
  $R := R - \frac{a}{b} X^{\deg R - \deg B} B$
Return (Q, R)

Remark 2.1

(1) If $\mathbb{K}$ is just a ring, the algorithm works if and only if B is monic.

(2) $A \equiv R \ \ mod \ \ B \quad \longrightarrow$ Euclidean division allows to perform operations in $\mathbb{K}[x]/(B)$.

$A_1 + A_2 \equiv R_1 + R_2 \ \ mod \ \ B$

$A_1 \times A_2 \equiv R_1 \times R_2 \ \ mod \ \ B$

2.3 Euclidean algorithm

Definition 2.6

R Euclidean domain and $a, b \in \mathbb{R}$, g is a gcd of a and b : $g = gcd(a, b)$

if : $g \in R$

$g|a$

$g|b$

any common divisor of $a$and$b$ divides g.

Proposition 2.3

In R Euclidean, such a gcd always exist.

Remark : g may not be unique.

Proposition : If $a = bq + r$ with $h(r) < h(b)$ then, $gcd(a, b) = gcd(b, r)$.

Algorithm 2: EuclidAlgorithm
Input: Two elements a and b in a Euclidean domain $\mathcal{R}$ with a height function h.
Output: A gcd of a and b in $\mathcal{R}$.
$r_0 := a, r_1 := b, i := 1$
While $r_i \neq 0$ **do**
    $r_{i+1} := \mathrm{rem}(r_{i-1}, r_i)$
    $i := i + 1$
Return r_{i-1}

Complexity : $O(deg(a) \times deg(b))$

Proposition 2.4

If $g = gcd(a, b)$ then, $\exists(u, v) \in R^2$ s.t $au + bv = g$ and $h(ug) < h(b), h(vg) < h(a)$

- u and v are cofactors.

Proposition 2.5

$a, b \in R$

EEA

Algorithm 3: ExtendedEuclideanAlgorithm
Input: Two elements a and b in a Euclidean domain $\mathcal{R}$ with a height function h.
Output: A gcd of a and b in $\mathcal{R}$ together with the corresponding cofactors.
$r_0 := a, u_0 := 1, v_0 := 0.$
$r_1 := b, u_1 := 0, v_1 := 1, i := 1$
While $r_i \neq 0$ **do**
    $(q_i, r_{i+1}) := \mathrm{PolynomialDivisionAlgorithm}(r_{i-1}, r_i)$
    $u_{i+1} = u_{i-1} - q_i u_i,\ v_{i+1} = v_{i-1} - q_i v_i$
    $i := i + 1$
Return $r_{i-1}, u_{i-1}, v_{i-1}$

Complexity : $O(deg(a) \times deg(b))$

Application of Extended Euclidean Algorithm: Modulo inversion

If a and b are coprimers, then $\exists(u, v)/au + bv = 1$.

So, $au \equiv 1 \mod b$ $bv \equiv 1 \mod a$

- u is a inverse of $a \mod b$

- v is a inverse of $b \mod a$

If $a \in \mathbb{Z}$ and $au + bv = 1$, then $a^{-1} \equiv u \mod n$

$\bar{a} = a + kn, K \in \mathbb{Z} \in \mathbb{Z}/n\mathbb{Z}, \bar{a}^{-1} = \bar{u} = u + kn, k \in \mathbb{Z}$

- If n is prime number then for all $a \in \mathbb{Z}, gcd(a, n) = 1$. So for all $\bar{a} \in \mathbb{Z}/n\mathbb{Z},\ \ \bar{a}^{-1}$ exists.

Proposition 2.6

$\mathbb{Z}/n\mathbb{Z}$ is a field if and only if n is prime.

Definition 2.7

$P \in \mathbb{K}[x]$ is irreducible if for any $\mathbb{Q}, \mathbb{R} \in \mathbb{K}[x]$ s.t $P = QR$, then either $\mathbb{Q} \in \mathbb{K}$ or $\mathbb{R} \in \mathbb{K}$.

Proposition 2.7

If P is irreducible then $\forall\, \mathbb{Q} \in \mathbb{K}[x],\quad gcd(P, Q) = 1$

Theorem 2.3

$\mathbb{K}[x]/(P)$ is a field if and only if P is irreducible

Remark 2.1

we can computer the inverse with Extended Euclidean Algorithm.

3 Finite field

Definition 3.1

A finite field is a field with a finite number of elements.

3.1 Integer $\mathbb{Z}$

- for $n \in \mathbb{Z}/0 \approx 0, ..., n-1$ with add/ multiplication modulo n

- $a \in \mathbb{Z}/n\mathbb{Z}$ is invertible if and only if $gcd(a, n) = 1$

- n is prime $\iff \quad \mathbb{Z}/n\mathbb{Z}$ is a field

- computing a^{-1} : run EEA to obtain $1 = au + nv$

Theorem 3.1: Bezout's relation

Let $R = \mathbb{Z} or R = \mathbb{K}[x]$. If a and b in R, there exist u and v in R s.t $au + bv = gcd(a, b)$

Theorem 3.2

Let $R = \mathbb{Z} or R = \mathbb{K}[x]$. If a and b are coprime, then a invertible modulo b and b invertible modulo a. Thus, $\mathbb{Z}/n\mathbb{Z}$ is a field, if and only if n is a prime.

3.2 Polynomial $\mathbb{K}[x]$

(where $\mathbb{K}$ is a field) - for $f \in \mathbb{K}[x] \setminus \{0\}$, $\mathbb{K}[x]/\langle f\rangle \approx \{P(X) \in \mathbb{K}[x]/deg(p) < deg(f)\}$ with add/multiplication mod f.

- $P \in \mathbb{K}[x]/\langle f\rangle$: P is invertible if and only if $gcd(p, f) = 1$

-f is invertible $\iff$ $\mathbb{K}[x]/\langle f \rangle$ field.

- computing P^{-1}: run EEA to obtain $1 = pu + fv$

- $\mathbb{K}[x]/\langle f \rangle$ is a field, for irreducible polynomial f.

Proof:

Suppose f irreducible, let $P \in \mathbb{K}[x]/\langle f \rangle \setminus \{0\}$. To show that P is invertible which means $gcd(P, f) = 1$. The gcd of P and f divides both P and f. But f has only $\mathbb{K}\{0\}$ and $\propto f, \propto \in \mathbb{K} \setminus \{0\}$ as divides. Since $deg(P) < deg(f)$, P cannot be divisible by f, so $gcd(P, f) = 1$.

3.3 Building a finite field

- If $K = \mathbb{Z}/p\mathbb{Z}$ and $deg(f) = d$ then $\mathbb{Z}/p\mathbb{Z}[x]/\langle f \rangle$ is a finite field of cardinality p^d. This is because $\mathbb{Z}/p\mathbb{Z}[x]/\langle f \rangle = \{a_0 + a_1 x + \ldots + a_{d-1}x^{d-1}, (a_0, a_1 \ldots.., a_{d-1}) \in (\mathbb{Z}/p\mathbb{Z})^d\}$ and cardinality of $(\mathbb{Z}/p\mathbb{Z}))^d$ is p^d

Theorem 3.3

A finite field must have p elements for same prime p and $d \in \{1, 2, \ldots..\}$. If $d = 1$ then finite field is $\mathbb{Z}/p\mathbb{Z}$. If $d > 1$, then $\mathbb{Z}/p\mathbb{Z}$ is not a field.

- $\mathbb{F}_q$ for $q = p^d$ is the notation for a finite field of cardinality q

III.6.

4 Formal Power Series

4.1 Introduction

Definition 4.1

From a sequence $(s_i) \in \mathbb{K}^{\mathbb{N}}$. We define the power series:

$$\sum_{i \in \mathbb{N}} s_i x^i$$

Proposition 4.1

- The set of power series is a ring which we write $\mathbb{K}[|x|]$

- Power series is an infinite sequence.

Examples:

$1 + x$ is a power series with coefficients $(1, 1, 0, 0, ..., 0, ..)$.

Remark 4.1

Polynomials are power series. $\iff$ (s_i) finitely many nonzero s_i

Operations $(+, \times)$ for power series

$$(1 - x) \sum_{i \in \mathbb{N}} x^i \; = \; 1 \quad \Rightarrow 1 \cdot \sum_{i \in \mathbb{N}} x^i \; - \; x \cdot \sum_{i \in \mathbb{N}} x^i$$

$$= (1, 1, 1, ..., 1, ...) \; - \; (0, 1, , 1, 1, ..., 1, ...) = (1, 0, 0, 0, ..., 0, ...)$$

- Addition: it is coefficient by coefficient.

$$\sum_{i\in\mathbb{N}} s_i x^i + \sum_{i\in\mathbb{N}} t_i x^i \;=\; \sum_{i\in\mathbb{N}} (s_i + t_i) x^i$$

- Multiplication :

$$(\sum_{i\in\mathbb{N}} s_i x^i)(\sum_{i\in\mathbb{N}} t_i x^i) \;=\; \sum_{i\in\mathbb{N}} (\sum_{k=0}^{i} s_k t_{i-k}) x^i$$

- 0 in $\mathbb{K}[|x|]$ is $(0 + 0x + 0x^2 + 0x^3 + ...)$

- 1 in $\mathbb{K}[|x|]$ is $(1 + 0x + 0x^2 + 0x^3 + ...)$

Remark 4.2

in $(\sum_{i\in\mathbb{N}} s_i x^i)$, x is not invertible. This can be proven by contradiction.

Examples 1 :

Fibonacci sequence $\longrightarrow$

$f_0 = 0, f_1 = 1, \quad \forall i, f_i = f_{i-1} + f_{i-2}$, then we can form:

$$S = \sum_{i\in\mathbb{N}} f_i x^i \in \mathbb{K}[|x|]$$

In this context with recurrent sequence, S is called the generating series of $(f_i)_{i\in\mathbb{N}}$.

Examples 2 :

Compute

$$(1 - x - x^2)S \;=\; \sum_{i\in\mathbb{N}} f_i x^i - \sum_{i\in\mathbb{N}} f_i x^{i+1} - \sum_{i\in\mathbb{N}} f_i x^{i+2}$$

$$= f_0 + f_1 x + \sum_{i\in\mathbb{N}} f_{i+2} x^{i+2} - (f_0 x + \sum_{i+1} x^{i+2}) - \sum_{i\in\mathbb{N}} f_i x^{i+2}$$

$$= f_0 + f_1 x - f_0 x + \sum_{i\in\mathbb{N}} (f_{i+2} - f_{i+1} - f_i) x^{i+2} = x$$

Sometimes, you can write a series in fractions,

$$S = \frac{x}{1 - x - x^2}$$

*it doesn't make sense to evaluate power series with specific value

some terminologies,

non-zero series : $[1, -1, -1, 0, 0, ..., 0, ..]$

zero series: $[0, 0, 0, 0, ..., 0..]$

infinite sequence: $\frac{1}{3} = 0.333333........$

$\mathbb{K}[x], \quad \{k(x) = \frac{p}{\phi}, \quad \phi \neq 0, \quad p, \phi \in \mathbb{K}[x]\}$

We work with power series:

- As fraction $\frac{p}{\phi}$ of two polynomials

- As a truncated power series at precision n:

$$S = s_0 + s_1 x + s_2 x^2 + ... + O(x^n)$$

where $O(x^n) : x^n T$ for some $T \in \mathbb{K}[|x|]$

4.2 Inversion

A power series S is invertible if there exists $T \in \mathbb{K}[|x|]$ such that $ST = 1$

Examples:

- $S = 0$ is not invertible

- $S = c \in \mathbb{K} \setminus \{0\} : S^{-1} = c^{-1}$

- $S = 1 - x - x^2$: invertible....why?

- $S = x$: not invertible....why?

- $S = 1 - x \quad : \quad (1 - x)^{-1} = \sum_{i \in \mathbb{N}} x^i$

Lemma 4.0

$S = \sum_{i \in \mathbb{N}} s_i x^i$ is invertible if and only if $S_0 \neq 0$

proof: Assume $S_0 \neq 0$. Construct $U = \sum_{i \in \mathbb{N}} u_i x^i \quad \text{s.t.} \quad US = 1$

Coefficient of degree $0 : 1 = u_0 S_0 \longrightarrow u_0 = S_0^{-1}$

Coefficient of degree $1 : 0 = u_0 S_1 + u_1 S_0 \longrightarrow u_1 = \frac{-u_0 S_1}{S_0}$

Coefficient of degree $2 : 0 = u_0 S_2 + u_1 S_1 + u_2 S_0$

Coefficient of degree $3 : 0 = u_0 S_3 + u_1 S_2 + u_2 S_1 + u_3 S_0$

.... continued

proceeding this way we get $u_2, u_3, ...$ defined uniquely.

From S at precision n, this gives $U = S^{-1}$ at precision n

Therefore, we can invert $S \in \mathbb{K}[|x|]$ known at precision n (with inverse at precision n) using $O(n^2)$ operations in $\mathbb{K}$.

Lemma 4.0

Let $S \in \mathbb{K}[[x]]$ be an invertible power series, and let $T = S^{-1} + O(x^n)$. Then the power series $U = T + (1 - TS)T$ satisfies $U = S^{-1} + O(x^{2n})$

Remark 4.1

For a differentiable function F, Newton's iteration for an approximated root x_k of $F(x) = 0$ is (ANUM!):

$$x_{k+1} = x_k - \frac{F(x_k)}{F'(x_k)}$$

In particular, for the case of power series inversion, power series S is the root of the function $F(x) = \frac{1}{x} - S$ so:

$$x_{k+1} = x_k - \frac{F(x_k)}{F'(x_k)} = x_k + (1 - x_k S)x_k$$

Also, notice that $U = S^{-1} + O(x^{2n})$ is the Newton's iteration applied to T

Algorithm 5: Power Series Inversion via Newton iteration
Input: An integer $n > 0$, and a truncated series $S = s_0 + \cdots + s_{n-1}x^{n-1} + O(x^n) \in \mathbb{K}[[x]]$ at precision n.
Output: The truncated power series U at precision n which satisfies $U = S^{-1} + O(x^n)$.
If $n = 1$ **then Return** s_0^{-1}
Compute recursively the inverse T of $S + O(x^{\lceil n/2 \rceil})$.
Return $U := T + (1 - TS)T + O(x^n)$.

Complexity analysis:

$f(n)$ = complexity of input precision n

$f(n) = 1$ if $n = 1$ and

$f(n) = f(\lceil \frac{n}{2} \rceil) + 2M(n) + 2n$

Roughly..... $f(n) = 2(M(n) + n) + 2(M(\frac{n}{2} + \frac{n}{2})) + 2(M(\frac{n}{4}) + \frac{n}{4}) + ... = O(M(n))$

Example 1 (Example IV 6)

With Newton's iteration, pay attention to $O(x^{\lceil n/2 \rceil}$, ignore all terms degree higher than $\lceil n/2 \rceil$.

Also, $\mathbb{F}_n[[x]]$, which indiecate that all computations are in $\mod n$,

$$S = 3 + 2x^2 + x^3 + x^7 \in \mathbb{F}_5[[x]]$$

$n = 8$ $(O(x^{\lceil n/2 \rceil})$

$$S = 3 + 2x^2 + x^3 + O(x^4)$$

$n = 4$

$$S = 3 + O(x^2)$$

$n = 1$

$$T = 3^{-1} + O(x) = 2 + O(x)(2 \text{ because } 3^{-1} \mod 5)$$

now algorithm returns.... (don't forget about $\mod 5$)

$$U = T + (1 - TS)T + O(x^2) = 2 + (1 - 2 \times 3)2 + O(x^2)$$

$$= 2 + (-5)2 + O(x^2) = 2 + O(x^2)$$

here, at $O(x^2), n = 4, S = 3 + O(x^2)$. On the next step $O(x^4)$, use S at $O(x^4)$ and previous computation of U.

$$V = U + (1 - US)U + O(x^4) = 2 + (1 - 2(3 + 2x^2 + x^3))2 + O(x^4)$$

$$= 2 + (-5 - 4x^2 = 2x^3)2 + O(x^4) = 2 + (x^2 + 3x^3)2 + O(x^4)$$

$$2 + 2x^2 + 6x^3 + O(x^4) = 2 + 2x^2 + x^3 + O(x^4)$$

The last step, using the previous computation of V

$$W = V + (1 - VS)V + O(x^8)$$

$$= 2 + 2x^2 + x^3 + (1 - (2 + 2x^2 + x^3)(3 + 2x^2 + x^3 + x^7))(2 + 2x^2 + x^3) + O(x^8)$$

*it is in precision of $O(x^8)$, thus you don't have to consider terms about x^8

$$= 2 + 2x^2 + x^3 + (1 - 6 - 10x - 5x^3 - 4x^4 - 4x^5 - x^6 - 2x^7)(2 + 2x^2 + x^3) + O(x^8)$$

now you have to remind yourself and consider that it is in $\mod 5$

$$= 2 + 2x^2 + x^3 + (x^4 + x^5 + 4x^6 + 3x^7)(2 + 2x^2 + x^3) + O(x^8)$$

$$= 2 + 2x^2 + x^3 + (2x^4 + 2x^5 + 8x^6 + 6x^7 + 2x^6 + 2x^7 + x^7) + O(x^8)$$

$$= 2 + 2x^2 + x^3 + 2x^4 + 2x^5 + 4x^7 + O(x^8)$$

4.3 Polynomial division with a reminder: fast algo.

The best algorithm known is based on Newton's inversion of power series.

Theorem 4.3

Given (A, B) polynomials of degree $m \geq n \geq 0$, we can compute a Euclidean division $A = BQ + R, \quad deg(R) < n$ in $O(M(m-n)) + M(n)$ operations in $\mathbb{K}$.

*division cost roughly the same as a multiplication for polynomials.

Idea of fast polynomial division:

- With two polynomials A with degree m and A with degree n, we want to find polynomials Q and R s.t. $A = BQ + R$, with $deg(R) < deg(B) = n$.
- $m \geq n$, otherwise the solution is $(Q, R) = (0, A)$ in other word $A = 0 + A$
- The idea of this algorithm is to exploit the gap between $deg(R)$ and $deg(A) = deg(BQ) = m$, thus gap $= m - deg(R) \geq m - n + 1$.

  We reverse the equality to put the gap in the low-degree coefficients:

  $x^m A(x^{-1}) = x^m B(x^{-1}) Q(x^{-1}) + x^m R(x^{-1})$
- Also, we can rewrite the solution of division as

$$\frac{A}{B} = Q + \frac{R}{B}$$

  Because $deg(R) < deg(B)$, with $x \longrightarrow \infty$, $\frac{R(x)}{B(x)} = 0$

Therefore Q corresponds to the asymptotic expansion of $\frac{A}{B}$ at infinity (∞). Hence, one can obtain Q by computing the Taylor expansion at infinity of the fraction $\frac{A}{B}$.

- To adapt above approach of an expansion at ∞, we can cange variable $y \leftarrow x^{-1}$, thus:

$$\frac{A(x^{-1})}{B(x^{-1})} = Q(x^{-1}) + \frac{R(x^{-1})}{B(x^{-1})}$$

  Then, multiply each side by x^{m-n} ($\frac{x^m}{x^n}$) to ensure that we only manipulate polynomials in numerators and denominators:

$$\frac{x^m A(x^{-1})}{x^n B(x^{-1})} = x^{m-n} Q(x^{-1}) + \frac{x^m R(x^{-1})}{x^n B(x^{-1})}$$

  Here, $deg(Q) = m - n$ and $x^n B(x^{-1})$ is invertible as a power series. (By assumption, B is nonzero.)

- Since $deg(R) < n$ and $m \geq n$, the polynomial $x^m R(x^{-1}) = x^{m-n+1} x^{n-1} R(x^{-1})$ has valuation at least $m - n + 1$ which is grater than the degree of polynomial $x^{m-n} Q(x^{-1})$. Thus expansion of $\frac{x^m A(x^{-1})}{x^n B(x^{-1})}$ at precision $m - n + 1$ will give us all coefficients of the polynomial $x^{m-n} Q(x^{-1})$, from which we can deduce to Q. Then $R = A - BQ$

FastPolynomialDivisionAlgorithm(A, B)

Input: Polynomials A and B in $\mathbb{K}[x]$ with B nonzero.

Output: Polynomials (Q, R) in $\mathbb{K}[x]$ such that $A = BQ + R$ and $\deg(R) < \deg(B)$.

1. Let $m = \deg(A)$ and $n = \deg(B)$
2. If $m < n$, return $(0, A)$
3. Compute the reversals $\tilde{A} = x^m\ A(1/x)$ and $\tilde{B} = x^n\ B(1/x)$
   (this step does not require any arithmetic operation in $\mathbb{K}$)
4. Compute $\tilde{Q} = \tilde{A}/\tilde{B} \mod x^{m-n+1}$ by inverting a formal power series and performing a power series multiplication, both at precision $m - n + 1$
5. Deduce Q by reverting the coefficients of $\tilde{Q}$
6. Deduce R by computing $A - BQ$
7. Return (Q, R)

$$A(x) = a_0 + a_1 x + 1_2 x^2 + \ldots + a_m x^m$$

$$A(1/x) = a_0 + \frac{a_1}{x} + \frac{a_2}{x^2} + \ldots + \frac{a_m}{x^m} \longrightarrow \tilde{A} = a_m + a_{m-1}x + \ldots + a_0 x^m$$

* multiplication of power series is same as polynomial multiplication

Example (Problem IV 9)

Compute a division of A/B where

$$A = 2 + x^3 + x^9 \quad B = 2 + 2x + x^3 \quad \text{in } \mathbb{F}_3[|x|]$$

Compute reversals $\tilde{A}$ and $\tilde{B}$. It is easier if you write coefficient on a array such that,

$$A = [2, 0, 0, 1, 0, 0, 0, 0, 0, 1] \quad B = [2, 2, 0, 1]$$

and then inverse the order.

$$\tilde{A} = 1 + x^6 + 2x^9 \quad \tilde{B} = 1 + 2x^2 + 2x^3$$

$m = 9, n = 3$, thus precision is $m - n - 1 = 7 \quad O(x^7)$

Compute $\tilde{Q} = \tilde{A}/\tilde{B} = \tilde{A}\tilde{B}^{-1}$. First, compute inverse of $\tilde{B} \rightarrow \tilde{B}^{-1}$

$$S = 1 + 2x^2 + 2x^3 + O(x^7)$$

$n = 7 \quad$ meaning $\; O(x^{\lceil 7/2 \rceil})$

$$S = 1 + 2x^2 + 2x^3 + O(x^4)$$

$n = 4$

$$S = 1 + O(x^2)$$

$n = 2$

$$S = 1 + O(x)$$

$n = 1$

$$S = 1^{-1} + O(x) = 1 + O(x)$$

Return $\tilde{B}^{-1}$

$$U = T + (1 - ST)T + O(x^2) = 1(1 - 1 \times 1)1 + O(x^2) = 1 + O(x^2)$$

$$V = U + (1 - US)U + O(x^4) = 1 + (1 - 1(1 + 2x + 2x^3)1 + O(x^4)) = 1 + (1 - 1 - 2x^2 - 3x^3)1 + O(x^4)$$

Reminder, computations are in $\mod 3$!!

$$T = V + (1 - VS)V + O(x^7) = 1 + x^2 + x^3 + (1 - (1 + x^2 + x^3)(1 + 2x^2 + 2x^3))(1 + x^2 + x^3) + O(x^7)$$

$$= 1 + x^2 + x^3 + (1 - (1 + 2x^2 + 2x^3 + x^2 + 2x^4 + 2x^5 + x^3 + 2x^5 + 2x^6))(1 + 2x^2 + 2x^3) + O(x^7)$$

$$= 1 + x^2 + x^3 + (1 - 1 - 3x^2 - 3x^3 - 2x^4 - 4x^5 - 2x^6)(1 + x^2 + x^3) + O(x^7)$$

$$= 1 + x^2 + x^3 + (x^4 + 2x^5 + x^6)(1 + 2x^2 + x^3) + O(x^7)$$

$$= 1 + x^2 + x^3 + x^4 + x^6 + 2x^5 + x^6 + O(x^7)$$

$$= 1 + x^2 + x^3 + x^4 + 2x^5 + 2x^6 + O(x^7)$$

Thus,

$$\tilde{B}^{-1} = 1 + x^2 + x^3 + x^4 + 2x^5 + 2x^6$$

now, compute $\tilde{A}\tilde{B}^{-1}$

$$\tilde{A}\tilde{B}^{-1} = (1 + x^6 + 2x^9)(1 + x^2 + x^3 + x^4 + 2x^5 + 2x^6)$$

$$= 1 + x^2 + x^3 + x^4 + 3x^6 = 1 + x^2 + x^3 + x^4 + 2x^5$$

Write down coefficient in array, $[1, 0, 1, 1, 1, 2]$ then deduce Q by reverting $\tilde{Q} = \tilde{A}\tilde{B}^{-1}$

$$Q = 2x + x^2 + x^3 + x^4 + x^6$$

Next, compute $R = A - BQ$

$$BQ = (x^3{+}2x{+}2)(2x{+}x^2{+}x^3{+}x^4{+}x^6) = 2x^4{+}x^5{+}x^6{+}x^2{+}2x^3{+}2x^4{+}2x^5{+}x{+}2x^2{+}2x^3{+}2x^4{+}2x^6{+}2x^7{+}x^7{+}x^9$$

$$= x + x^3 + x^9$$

$$R = (x^9 + x^3 + 2) - (x^9 + x^3 + x) = 2 - x = 2x + 2$$

Final step is necessary because $\mathbb{F}_3[|x|]$. The solution is :

$$(Q, R) = (2x + x^2 + x^3 + x^4 + x^6, 2x + 2)$$

5 Fast Polynomial Evaluation and Interpolation

5.1 Introduction

There are two main questions we want to solve in this chapter.

Question 1: (Multipoint evaluation)

- Input: n elements $x_0, ..., x_{n-1} \in \mathbb{K}$
- Input: Polynoimial $P = p_{n-1}x^{n-1} + \ldots + p_0 \in \mathbb{K}[x]$ of degree less than n.
- How to efficiently compute $y_i = P(x_i)$ for $0 \leq i < n$?

Question 2 (Interpolation)

- Input: n pairwise distinct element $x_0, ..., x_{n-1} \in \mathbb{K}$
- Input: Polynoimial $P = p_{n-1}x^{n-1} + \ldots + p_0 \in \mathbb{K}[x]$ of degree less than n.
- How to efficiently compute a polynomial $P = p_{n-1}x^P n - 1 + \ldots + p_0 \in \mathbb{K}[x]$ s.t. $P(x_i) = y_i$ for all $0 \leq i < n$

*Those two questions are inverse to each other

5.1.1 Multipoint evaluation

The very naive algorithms take $2n$ multiplications and n addition. This should never be used. Instead of this algorithm, a less naive algorithm which is the Horner scheme is recommended. This take n multiplications and n addition.

Horner scheme

⋮

5.1.2 Interpolation

The first approach for interpolation is to rely on the Lagrange. In conclusion, it takes $O(n^2)$ in $\mathbb{K}$.

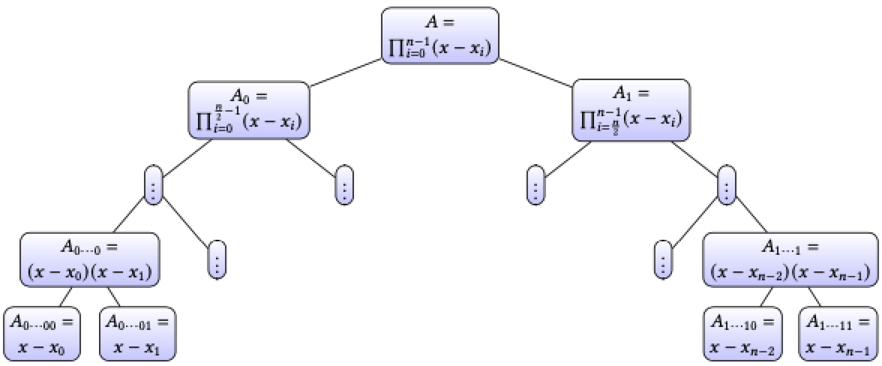
Lagrange interpolation : https://youtu.be/WCGKqJrf4N4

5.2 Fast multipoint evaluation

Computing $y_i = P(x_i)$ for $0 \leq i < n \quad \longrightarrow P \mod (x - x_i)$ for $0 \leq i < n$. We consider $n = 2^k$ for simplicity.

- First, compute $A = \prod_{i=0}^{n-1}(x - x_i)$ and a corresponding subproducts tree.
- Second, compute $P \mod (x - x_i)$ for $0 \leq i < n$ by exploiting the subproducts tree

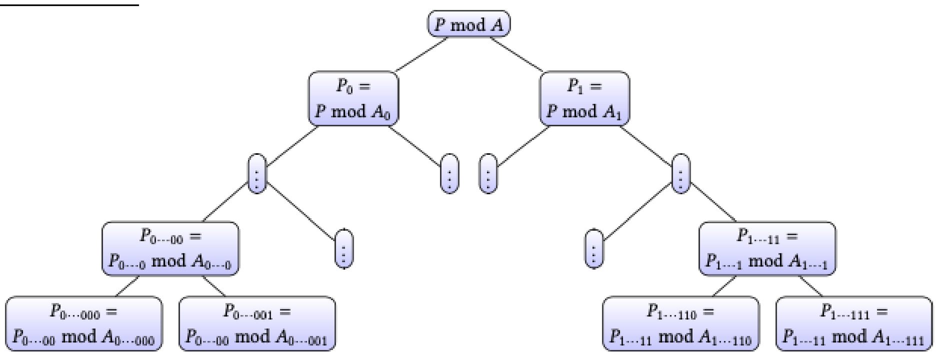
subproducts tree



Proposition 5.1

Building the subproducts tree from its leaves to to its root yield all the polynomials indicated in its nodes in a total of $O(M(n)log(n))$ operation in $\mathbb{K}$

<u>reminders tree</u>



Proposition 5.2

Recall that we assume $deg(P) < n$. Having computed the suibproducts tree computing all the remainders in the remainders tree from its root to its leaves uses $O(M(n)log(n))$ operation in $\mathbb{K}$

Algorithm 6: Fast multipoint evaluation
Input: An integer $n > 0$, a polynomial $P \in \mathbb{K}[x]$ of degree $< n$, and $x_0, \ldots, x_{n-1}$ in $\mathbb{K}$.
Output: $P(x_0), \ldots, P(x_{n-1})$.
Compute the subproducts tree of $x_0, \ldots, x_{n-1}$.
Compute the remainders tree of P with respect $x_0, \ldots, x_{n-1}$, using the subproducts tree.
Return the remainders at the leaves of the remainders tree.

Theorem 5.1

Fast polynomial multipoint evaluation can be performed in $O(M(n)log(n))$ operations in $\mathbb{K}$.

5.3 Fast Interpolation

We use divide and conquer approach. Recall form Lagrange formula, poynomial P

$$P(x) = \sum_{i=0}^{n-1} (y_i / L_i(x_i)) L_i x = A(x) \sum_{i=0}^{n-1} \frac{y_i / L_i(x_i)}{x - x_i}$$

from the formula above, P is exactly the numerator of a sum of fractions of the form

$$S = \sum_{i=0}^{n-1} \frac{c_i}{x - x_i} \text{ for some fixed elements } c_0, \ldots, c_{n-1} \in \mathbb{K} \;\; c_i = \sum_{i=0}^{n-1} \frac{y_i}{L_i(x_i)}$$

Proposition 5.1

For all $0 \leq i < n$, we have $L_i(x_i) = A'(x_i)$ where A' is the derivative of A

6 Guessing Linear Recurrence Relations

6.1 The Berlekamp-Massey algorithm

Definition 6.1

A sequence $b = (b_i)_{i\in\mathbb{N}}$ with terms in a field $\mathbb{K}$ is said to be linearly recurrent if there exists $d \in \mathbb{N}$ and $v_0, \dots v_{d-1} \in \mathbb{K}$ s.t.

$$\forall i \in \mathbb{N}\,, \quad b_{i+d} + v_{d-1}b_{i+d-1} + \dots + v_0 b_i = 0$$

Any such integer $d \in \mathbb{N}$ is called an order of the linear recurrence.

Proposition 6.1

A sequence $b = (b_i)_{i\in\mathbb{N}}$ with terms in a field $\mathbb{K}$ is said to be linearly recurrent if there exists $d \in \mathbb{N}$ and $v_0, \dots v_{d-1} \in \mathbb{K}$ s.t.

$$\forall i \in \mathbb{N}\,, \quad b_{i+d} + v_{d-1}b_{i+d-1} + \dots + v_0 b_i = 0$$

Any such integer $d \in \mathbb{N}$ is called an order of the linear recurrence.

Remark 6.1

For a sequence b whose terms are not given by a formula, testing that a linear recurrence relation is satisfied by d is not possible.

Guessing a linear recurrence relation satisfied by b is the problem of computing a suitable linear recurence relation based on finitely many terms of b.

Consider only first D terms since there is infinite terms. If D is significantly huge, then one can hope to found linear recurrence hold for whole infinite sequence. In practical case, one usually knows a bound on the order of the recurrence of the considered sequence which allows to select suitable D which ensure that a recurrence for the first D terms actually gives a recurrence for the whole sequence.

6.1.1 Algorithm

Computing a linear recurrence relation of order d satisfied by $b = (b_i)_{i\in\mathbb{N}}$ from its first D terms comes down to finding $v_0, \ldots, v_{d-1}$ s.t

$$\begin{cases} v_0 b_0 + \ldots + v_{d-1} b_{d-1} + b_d = 0 \\ v_0 b_1 + \ldots + v_{d-1} b_d + b_{d+1} = 0 \\ \vdots \\ v_0 b_{D-1-d} + \ldots + v_{d-1} b_{D-2} + b_{D-1} = 0 \end{cases}$$

thus, we look for the smallest $d \in 0, \ldots, D-1$ s.t. the Hankle matrix

$$\begin{pmatrix} b_0 & \ldots & b_{d-1} & b_d \\ b_1 & \ldots & b_d & b_{d+1} \\ \vdots & & \vdots & \vdots \\ b_{D-1-d} & \ldots & b_{D-2} & b_{D-1} \end{pmatrix}$$

has a right kernael which contains a vector of the form $\begin{pmatrix} v_0 \\ \vdots \\ v_{d-1} \\ 1 \end{pmatrix}$

the matrix-vector product

$$\begin{pmatrix} b_0 & \dots & b_{d-1} & b_d \\ b_1 & \dots & b_d & b_{d+1} \\ \vdots & & \vdots & \vdots \\ b_{D-1-d} & \dots & b_{D-2} & b_{D-1} \end{pmatrix} \begin{pmatrix} v_0 \\ \vdots \\ v_{d-1} \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

can be extended into

$$\begin{pmatrix} b_0 & \dots & b_{d-1} & b_d \\ b_1 & \dots & b_d & b_{d+1} \\ \vdots & & \vdots & \vdots \\ b_{D-1-d} & \dots & b_{D-2} & b_{D-1} \\ b_{D-d} & \dots & b_{D-1} & 0 \\ \vdots & \iddots & \iddots & \vdots \\ b_{D-1} & 0 & \dots & 0 \end{pmatrix} \begin{pmatrix} v_0 \\ \vdots \\ v_{d-1} \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ r_{d-1} \\ \vdots \\ r_0 \end{pmatrix}$$

with $r_0, \dots, r_{d-1}$ unknown. This product corresponds to the polynomial product

$$B_D = \sum_{i=0}^{D-1} b_i X^{D-1-i} \text{ and } V = X_d + \sum_{i=0}^{d-1} v_i X^i \mod X^D$$

and,

$$B_D V = r_0 + r_1 x + \ldots + r_{d-1} x^{d-1} \mod X^D$$

This means we look for U and V using EEA s.t.

$$x^D U + B_D V = R, \quad deg(R) < deg(V)$$

R corresponds to coefficients of the recurrence. Look for a reminders that is of degree $deg(R) < deg(V)$. Thus we perform EEA until $deg(R) < deg(V)$. *Here, R and V refers to R and V in EEA algorithm

Algorithm:

In: D terms of a sequence $b = (b_i)_{i\in\mathbb{N}}$

Out: $v = (v_i)_{0\leq i\leq d} \quad d \leq D/2$

satisfying : $EEA(X^D, B_D)$ stop when $deg(R) < deg(V)$

Historically, there exist many different description on Berlekamp-Massay algorithm. This viewpoint allows to link to extended Euclidean algorithm.

Example

Given that: $\mathbb{F}_{13}$. $d = 2$, $D = 4$ and ...

$b = (6, 7, 7, 1)$

$$B_D = \sum_{i=0}^{D-1} b_i x^{D-i-1} \quad \text{so} B_D = 1 + 7x + 7x^2 + 6x^3$$

Now, we do $EEA(X^D, B^D)$ which in this case $EEA(X^4, B^D)$

Then, $X^D u + B_D v = R$ and $B_D V = r_0 + r_1 x + \ldots + r_{d-1} x^{d-1} \mod X^D$. After perform $EEA(X^4, B^D)$ we have that $V = x^d + \sum_{i=0}^{d-1} v_i x^i = 2 + 9x + 7x^2$.

Here, we assume leading coefficient is 1, so $V/7$.

$$V = x^2 + 5x + 4$$

Theorem 6.1

The Berlekamp-Massey algorithm performs the guessing of a linear recurrence from the first D terms of a sequence using $O(D^2)$ operations in $\mathbb{K}$. Using algorithms elaborating upon the fast extended Euclidean algorithm, this problem can be solved in complexity $O(M(D) \log D)$

6.2 Sparse Matrix

Definition 6.1

A sequence $b = (b_i)_{i \in \mathbb{N}}$ over $\mathbb{K}$ is linearly recurrent of order d of smallest relation

$$b_{i+d} + v_{d-1} b_{i+d-1} + \ldots + v_0 b_i = 0$$

if, and only if, there exists a polynomial $R \in \mathbb{K}[x]$ s.t $deg(R), d$ and

$$\sum_{i=0}^{\infty} b_i x^i = \frac{R}{1 + v_{d-1} x + \ldots + v_0 x^d}$$

Definition 6.2: Sparse Matrix

A sparse matrix is a matrix with many (most) coefficients that are zero.

*There are no specific numbers of zero to be sparse matrix

You can possibly compute matrix computations faster by taking into account that a matrix is sparse.

Data representation

You can use i, j coefficients for all non-zero coefficients.

For example, $M = \begin{bmatrix} \alpha_1 & 0 & \dots & 0 \\ \alpha_2 & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \alpha_n & 0 & \dots & 0 \end{bmatrix}, \quad \begin{matrix} 0, 0, \alpha_1 \\ 1, 0, \alpha_2 \\ 2, 0, \alpha_3 \\ \vdots \\ n-1, 0, \alpha_n \end{matrix} \quad , \longrightarrow$ representation of size $O(n)$

6.2.1 Multiplication

Product of two sparse matrices is not necessary sparse.

Given a matrix with m nonzero entries, stored with a sparse representation, its product with a vector requires $O(m)$ operations in the base field. Recall that for $n \times n$ dense matrix, multiplication can be done $O(n^{2.38})$ with Coppersmith–Winograd algorithm.

Given two matrices A and B, with both size of $n \times n$ and we assume that the number of non-zero entries in those matrices are at most m_A, m_B. The naive algorithm of matrix multiplication can take some advantage of sparsity since the product of any element from A and B will be zero if either the corresponding entry in A or B is zero. Let, $\overline{a}_k$ and $\overline{b}_k$ be a number of non-zeror element in kth col (resp. row) of matrix A and B. In particular,

$$\sum_{k=1}^{n} \overline{a}_k = m_A \text{ and } \sum_{k=1}^{n} \overline{b}_k = m_B$$

Then the number of multiplications in the base ring that one does in the naive matrix multiplication AB is

$$\sum_{k=1}^{n} \overline{a}_k \overline{b}_k \leq (\sum_{k=1}^{n} \overline{a}_k)(\sum_{k=1}^{n} \overline{b}_k) \leq m_A m_B$$

Note that one should also consider the number of additions, and that the details of such a naive sparse matrix multiplication algorithm will be highly dependent on the chosen sparse representation format.

6.2.2 Other computation

Addition

$A, B \in \mathbb{K}^{n\times n}$ with m_A, m_B non-zero entries.

$A+B$ in $O(m_A+m_B)$ operations and $A+B$ has $\leq m_A+m_B$ non-zero entries.

Matrix-vector multiplication

$A \in \mathbb{K}^{n\times n}$ with m_A non-zero entries, and $V \in \mathbb{K}^n$ (dense). AV in $O(m_A)$ operations.

Computing multiplication of dense and sparce matrices

$A \in \mathbb{K}^{n\times n}$ with m_A non-zero entries, and $B \in \mathbb{K}^{n\times n}$ (dense). Multiplication can be done in $O(m_a, n)$, faster than multiplication of dense matrices.

PLU decomposition

A sparse matrix can have a dense PLU decomposition. Example on the lecture note.

6.2.3 The Wiedemann algorithm

goal : efficiently computer a nonzero vector in the kernel of a matrix.

it is probabilistic algorithm

let . . .

- $M \in \mathbb{K}^{m\times n}$ is a sparse matrix
- x_0 randomly picked vector of $\mathbb{K}^n$
- $x = Mx_0$
- randomly picked vector $y \in \mathbb{K}^n$

The matrix M has a minimal polynomial

$$P = z^r + p_{r-1}z^{r-1} + \ldots + p_0 \quad \text{of degree} \quad r \leq n$$

that is there exist $r \in \mathbb{N}$ minimal and $p_0, \ldots p_{r-1} \in \mathbb{K}$ s.t.

$$M^r + p_{r-1}M^{r-1} + \ldots + p_0 Id = 0$$

Multiplying on the right this equality by x, we obtain

$$s_r + p_{r-1}s_{r-1} + \ldots + p_0 s_0 = 0, \;\; s_i = M^i x$$

$$\rightarrow s = (s_i)_{i \in \mathbb{N}} = (x, Mx, m^2 x, \ldots) = (M^i x)_{i \in \mathbb{N}}$$

The terms of this vector sequence are computed recursively

$$s_0 = x \quad \text{and for all} \quad i \in \mathbb{N}, \;\; s_{i+1} = M s_i$$

since $x = Mx_0$ then

$$s_r + p_{r-1}s_{r-1} + \ldots + p_0 s_0 = M(M^r x_0 + p_{r-1}M^{r-1}x_0 + \ldots + p_0 x_0) = 0$$

Hence, $M^r x_0 + p_{r-1}M^{r-1}x_0 + \ldots + p_0 x_0$ is a vector in the kernel of M. This vector is the zero one.

Assume there exist $d \in \mathbb{N}$ s.t. $d < r$ and $q_0, \ldots, q_{d-1} \in \mathbb{K}$ s.t.

$$s_r + p_{r-1}s_{r-1} + \ldots + p_0 s_0 = M(M^r x_0 + p_{r-1}M^{r-1}x_0 + \ldots + p_0 x_0) = 0$$

$$M^r x_0 + p_{r-1}M^{r-1}x_0 + \ldots + p_0 x_0 \neq 0$$

then $M^r x_0 + p_{r-1}M^{r-1}x_0 + \ldots + p_0 x_0$ is nonzero vector in the kernel of M

Idea is to compute these $q_0, \ldots, q_{d-1}$ by multiplying first this equation on the left by $y^T M^i$ for all i. Thus

$$\forall i \in \mathbb{N} \quad b_{i+d} + q_{d-1}b_{i+d-1} + \ldots + q_0 b_i = 0$$

where $b_i = y^T M^i x$ and $b = (b_i)_{i \in \mathbb{N}} = (y^T x, y^T Mx, y^T M^2 x \ldots) = (y^T M^i x)_{i \in \mathbb{N}}$. These sequence terms can be computed as

$$b_i = y^T s_i$$

In other words the sequence b satisfies the linear recurrence relation for all i, $b_{i+d} + q_{d-1} b_{i+d-1} + \ldots + q_0 b_i = 0$. To compute these d and $q_0, \ldots, q_{d-1}$, we use Berlekamp-Massay algorithm which determine the linear recurrence relation of smallest order satisfied by the sequence.

Theorem 6.2

The Wiedemann algorithm, called on an $n \times n$ sparse matrix with at most m nonzero entries, $m \leq n$ uses $O(mn)$ operations in the vase field.

Building the sequence is the botleneck of the algorithm.
Determining the linear recurrence relation uses $O(M(n) \log n)$ operations in the base field. One can take $M(n)$ to be quasi-linear in n, so that this is in $O(n^2)$, hence this costs $O(nm)$ field operations (in fact, even the Karatsuba algorithm is enough for $O(M(n) \log n)$ to be in $O(n^2)$; however, the naive quadratic algorithm would not be suitable).

7 Univariate and Bivariate Results

Consider

- Univariate resultants when the ring is a field $R = \mathbb{K}$
- Bivariate resultants where the ring is the univariate polynomial $R = \mathbb{K}[x]$

7.1 Definition: Sylvester matrix, resulant

let $A, B \in R[x]$ be two polynomials

$$A = a_m x^m + \ldots + a_0 \quad B = b_n x^n + \ldots + b_0 \quad a_m \neq 0, b_n \neq 0$$

Sylvester matrix (A, B), $(m+n) \times (m+n)$ matrix over R

$$Syl(A, B) = \begin{bmatrix} a_m & \ldots & a_1 & a_0 & & \\ & \ddots & & & \ddots & \\ & & a_m & a_{m-1} & \ldots & a_0 \\ b_n & \ldots & b_1 & b_0 & & \\ & \ddots & & & \ddots & \\ & & b_n & b_{n-1} & \ldots & b_0 \end{bmatrix} \in R^{(m+n)\times(m+n)}$$

first n rows : coefficient vectors of the first n shifts of $A \longrightarrow x^{n-1}A, \ldots, xA, A$.

first m rows : coefficient vectors of the first m shifts of $B \longrightarrow x^{n-1}B, \ldots, xB, B$.

This matrix has a particular strictures, its $(m+x)^2$ entries are defined from only $m+n$ elements of R.

- elements are coefficients of polynomials A and B

- because of this structure, vector-matrix products $W\ Syl(A, B) \quad W \in R^{1\times(m+n)}$

$U, V \in R[x]$, two polynomials, $deg(U) < deg(B) = n \quad deg(V) < deg(A) = m$ then we can define $W = [u_{n-1} \dots u_1, u_0, v_{m-1}, \dots, v_1, v_0] \in R^{1\times(m+n)}$

$$W\ Syl(A, B) = (u_{n-1}x^{n-1}A + \dots u_1xA + u_0A) + (v_{m-1}x^{m-1}B + \dots v_1xB + v_0B) = AU + BV$$

R-linear combinations of the rows of $Syl(A, B)$ allow us to represent all such polynomial combinations $AU + BV$ when we restrict $deg(U) < n$ and $deg(V) < m$

If $R = \mathbb{K}$ then the Bezont relations tells us that one of these combinations yield the GCD of A and B, that is $AU + BV = gcd(A, B)$. This GCD is th polynomial of smallest degree which can be obtain as such a polynomial combination. The vector of coefficients of the GCD can be retrieved as the last nonzero row in a row echelon form of $Syl(A, B)$.

Definition 7.1

Given nonzero polynomials A and B in $R[x]$, the resultant of (A, B) is the determinant of the Sylvester matrix $Syl(A, B)$. We denote it by $Res_x(A, B)$. It is an element of R.

Note* that $Res_x(A, B) \neq Res_x(B, A)$

Lemma 7.0

$R = \mathbb{K}$. Let A and B be nonzero polynomials in $\mathbb{K}[x]$. Then, A and B are coprime if and only if $Res_x(A, B) \neq 0$

7.2 Properties of the resultant

Theorem 7.2: Poisson's formula

Assume that the polynomials A and B in $R[x]$ factorize as

$$A = a(x - \alpha_1)\dots(x - \alpha_m) \quad \text{and} \quad B = b(x - \beta_1)\dots(x - \beta_n)$$

where, $a, b, \alpha_i, \beta_j \in R$. Then resultant of (A, B) is

$$Res_x(A, B) = a^n b^m \prod_{i,j}(\aleph_i - \beta_j) = (-1)^{mn} b^m \prod_{1 \le j \le n} A(\beta_j) = a^n \prod_{1 \le i \le m} B(\alpha_i) = (-1)^{mn} Res_x(B, A)$$

The resultant is multiplicative: for nonzero polynomials A, B, C in $R[x]$

$$Res_x(AB, C) = Res_x(A, C) Res_x(B, C)$$

Proposition 7.1

For nonzero polynomials A and B in $R[x]$, there exist U and V in $R[x]$ such that $Res_x(A, B) = AU + BV$ and $deg(U) < n = deg(B)$ and $deg(V) < m = deg(A)$

Remark 7.1

Let R and R' be two commutative rings, and $\varphi : R \longrightarrow R'$ be a ring homomorphism. We extend φ into a polynomial ring homomorphism $\varphi : R[x] \longrightarrow R'[x]$, in the natural way by setting $\varphi(x) = x$. Let A and B be nonzero polynomials in $R[x]$, of respective degrees m and n.

- If $deg(\varphi(A)) = m$ and $deg(\varphi(B)) = n$ then
  $\varphi(Res_x(A, B)) = Res_x(\varphi(A), \varphi(B))$.
- If $deg(\varphi(A)) = m$ and $deg(\varphi(B)) = n' < n$ then

$\varphi(Res_x(A,B)) = \varphi(f_m)^{n-n'} Res_x(\varphi(A), \varphi(B))$.

- $deg(\varphi(A)) = m' < m$ and $deg(\varphi(B)) = n$ then
  $\varphi(Res_x(A,B)) = (-1)^{(m-m')n} \varphi(g_n)^{m-m'} Res_x(\varphi(A), \varphi(B))$.
- if $deg(\varphi(A)) < m$ and $deg(\varphi(B)) < n$ then $\varphi(Res_x(A,B)) = 0$ but nothing can be said in general about $Res_x(\varphi(A), \varphi(B))$

7.3 Univariate resultant: algorithms

Naive way to compute $Res_x(A,B)$ is linear algebra, find determinant of the Sylvester matrix with Gaussian elimination $\rightarrow O((m+n)^3)$ operations in $\mathbb{K}$

Faster algorithm reduces the determinant of matrix to matrix multiplication, $O((m+n)^{2.81})$ operations with Strassen's multiplication and $O((m+n)^{2.38})$ with current best known algorithm.

Above algorithm do not take into account of Sylvester matrix characteristics. Considering those charactristics. it can be comupted much faster.

Theorem 7.3

Let A and B nonzero polynomials in $\mathbb{K}[x]$. Let Q and R be polynomials in $\mathbb{K}[x]$ s.t. $A = BQ + R$ and $r = deg(R) < deg(B) = n$. Then,

$$Res_x(A,B) = (-1)^{mn} b_n^{m-r} Res_x(B,R)$$

where b_n is the leading coefficient of B

7.4 Bivariate resultant: algorithm

Consider the case of the bivariate resultant $R = \mathbb{K}[x]$, coefficient of A and B are univariate polynomial; and the Sylvester matrix is $(m+n) \times (m+n)$ with entries in $\mathbb{K}[y]$. Finding determinant in usually way with linear algebra is not appropriate, aldo Gaussian elimination is not suited either since $\mathbb{K}[y]$ is not field.

- not allowed to divide by a non-constant polynomial (otherwise we get fractions which are not in $\mathbb{K}[y]$) anymore)
- even we accept, it leads to numerators and denominators having very large degree

problem 2 $\vdots$

7.5 Bivariate resultant: solving bivariate polynomial systems

A and B, two polynomials in $\mathbb{K}[x.y]$ as usual. Assume field $\mathbb{K}$ is closed, find all points $(\alpha, \beta) \in \mathbb{K}^2$ such that $A(\alpha, \beta) = B(\alpha, \beta) = 0$ $A = 0$ and $B = 0$ are described as curve on $\mathbb{K}^2$, so solving system $S : A = B = 0$ id equivalent to finding the intersection points of two curve.

Idea is eliminate a variables, thus reducing the problem to finding root of univasriate polynomials.

- 1. We consider the system S_x , which is the same as S except that A and B are seen as univariate polynomials in x over $R = \mathbb{K}[y]$, that is, $A, B \in \mathbb{K}[y][x]$. Solving S_x means finding the common "x roots" of both polynomials.
- 2. We know that A and B have a common root if their resultant is 0; this resultant is a polynomial in A and y,indeed we have seen $Res_x(A, B) \in R = \mathbb{K}[y]$. (Here,we see the importance of x in the notation $Res_x(A, B)$: it indicates we eliminate the variable x .)
- 3. Since $Res_x(A, B) = AU + BV$ for some $U, V \in R[x]$,if (α_0, β_0) is a solution of S ,then $Res_x(A, B)$ evaluated at $y = \beta_0$ vanishes. In other words, $A(x, \beta_0)$ and $B(x, \beta_0)$, seen as polynomials in x , have at least one common root α_0 . Thus, their resultant is 0 and β_0 is

a zero of $Res_x(A, B)$.

Beware: $Res_x(A, B)$ can vanish at some β_{-1} without the existence of a solution $(\alpha_{-1}, \beta_{-1})$ of the system S! Such a root of $Res_x(A, B)$ is called a parasite solution, or extra solution.

- 4. For each root β_i of $Res_x(A, B)$, we find the list of α_i' s such that $A(\alpha_i, \beta_i) = B(\alpha_i, \beta_i) = 0$.

8 Structured linear algebra

8.1 Introduction and definitions

Remember that $n \times n$ matrix with entries in field k can be manipulated (multiplication, inversion, system solving, etc) in $O(n^{\omega})$ where ω is real number between 2 and 3.

There are some different matrix structures that are defined.

- Toeplitz
- Hankel
- Vandermonde
- Cauchy

Definition 8.1

A matrix $A \in \mathbb{K}^{n \times n}$ is said to be Toeplitz matrix if it is invariant along the diagonals, meaning that its entries $a_{i,j}$ satisfy $a_{i,j} = a_{i+k,j+k}$ for all k

Remark that a Toeplitz matrix is entirely described by its first row and its first column, that is, by only $2n - 1$ coefficients from $\mathbb{K}$.

All diagonal coefficients are equal.

Example

$$\begin{bmatrix} a_0 & a_{-1} & a_{-2} & \cdots & \cdots & a_{-(n-1)} \\ a_1 & a_0 & a_{-1} & \ddots & & \vdots \\ a_2 & a_1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & a_{-1} & a_{-2} \\ \vdots & & \ddots & a_1 & a_2 & a_{-1} \\ a_{n-1} & \cdots & \cdots & a_2 & a_1 & a_0 \end{bmatrix}$$

Definition 8.2

A matrix $A \in \mathbb{K}^{n \times n}$ is said to be Hankel matrix if it is invariant along the antidiagonals, meaning that its entries $a_{i,j}$ satisfy $a_{i,j} = a_{i-k,j+k}$ for all k

Thus, a Hankel matrix is entirely described by its first row and its last column, that is, again by only $2n - 1$ coefficients from $\mathbb{K}$

All antidiagonal coefficients are equal.

Example

$$\begin{bmatrix} a & b & c & d & e \\ b & c & d & e & f \\ c & d & e & f & g \\ d & e & f & g & h \end{bmatrix}$$

Lemma 8.0

Multiplying a Toeplitz matrix (or Hankel matrix) in $\mathbb{K}^{n \times n}$ by a vector in $\mathbb{K}^{n \times 1}$ can be done in $O(M(n))$ operations in $\mathbb{K}$

Definition 8.1

A matrix $A = (a_{i,j})_{i,j=0}^{n-1}$ in $\mathbb{K}^{n\times n}$ is a <u>Vandermonde</u> matrix if its entries can be written $a_{i,j} = a_i^j$ for $a_0, \ldots, a_{n-1} \in \mathbb{K}$.

Definition 8.2

A matrix $A = (a_{i,j})_{i,j=0}^{n-1}$ in $\mathbb{K}^{n\times n}$ is a <u>Cauchy</u> matrix if its entries can be written $a_{i,j} = 1/(a_i - b_j)$ for $a_i, b_j \in \mathbb{K}$, with $a_i \neq b_j$ for all i, j

From above, the matrix can be represented by only $O(n)$ elements from the field $\mathbb{K}$. Matrix-vector product can be computed more efficiently than for arbitrary dense matrices. The matrix-matrix product with another $n \times n$ can be computed more efficiently.

Matrix-vector, matrix-matrix can be computed in time quasi-optimal in the size of output using FFT-based polynomial multiplications.

Matrix-vector Av has a polynomial interpretations:

- polynomial multiplication for Toeplitz, Hankel, and Sylvester matrices
- multipoint evaluation of polynomials for Vandermonde matrices
- for Cauchy matrices, multipoint evaluation of rational fractions in $\mathbb{K}(x)$ of the form $\sum_{j=0}^{n-1} c_j/(x - b_j)$

The system-solving can also have a polynomial interpretations:

- guessing linear recurrences with constant coefficients, for Toeplitz or Hankel matrices
- interpolation of polynomials, for Vandermonde matrices
- interpolation of rational fractions, for Cauchy matrices

above three operations with quasi-optimal algorithm with $O(M(n)log(n)$

If A is an invertible matrix such that Av can be computed in L operations in $\mathbb{K}$, there is no known way to deduce existence of algorithm with complexity $O(L)$ for computing $A^{-1}v$

Structure matrix is not stable by inversion. For instance, inversion of Toeplitz is not Toeplitz

Displacement operator

$$\phi(A) = A - (\text{A shifted by 1 to down and 1 to the right})$$

If A is Toeplitz matrix, the $\phi(A)$ has a square submatrix of dimention $(n-1) \times (n-1)$ with zero entrties. In particular this matrix has rank at most 2. We say that ϕ is a displacement operator, and that A has a displacement rank (with respect to ϕ) which is at most 2.

Definition 8.3

The matrix G, H in $\mathbb{K}^{n\times 2}$ such that $\phi(A) = GH^T$ are called displacement generators for the Toeplitz matrix A

Definition 8.4

The displacement operatos ϕ_+ is $\phi(A) = A - ZAZ^T$ where $A \in \mathbb{K}^{n\times n}$ and Z is the shift matrix defined as

$$Z = \begin{pmatrix} 0 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ \vdots & \ddots & \dots & \vdots \\ 0 & \dots & 1 & 0 \end{pmatrix} \in \mathbb{K}^{n\times n}$$

(Thus ZA) is the matrix A shifted by one row to the bottom, and AZ^T is the matrix A shifted by one column to the right. Hence, this definition coincides with the above definition for Toeplitz matrices.)

A displacement rank of A is the integer $\alpha(A) = rank(\phi(A)$. Then there exists a pair of $n \times \alpha$ matrices (G, H) such that $\phi(A) = GH^T$. Any such pair (G, H) is called a displacement generator (with respect to the operator ϕ). If $\alpha(A) << n$, we say that A is quasi-Toeplitz, and the integers $\alpha(A)$ is also called the length of the generator (G, H)

8.2 main result

The key idea behind fast algo. for structures matrices with displacement rank α is to use the displacement generators as a concise data structure. Indeed it has size $O(\alpha n)$ linear in the size n of the matrix when $\alpha << n$

Theorem 8.2

Let $A \in \mathbb{K}^{n\times n}$ be a quasi-Toeplitz matrix with displacement rank α , given by generators $G, H \in \mathbb{K}^{n\times\alpha}$. Then one can compute

- the determinant of A
- the rank of A
- a linear system solving $Ax = b$

in $O\tilde{}(\alpha^2 n)$ operation in $\mathbb{K}$

$O(\alpha^2 M(n)log(n))$ in the quais-Toeplitz case, and $O(\alpha^2 M(n)log^2(n))$ in the quasi-Vandermonde and quasi-Cauchy cases.

Consequences

- The extended GCD of polynomials $f, g \in \mathbb{K}[x]$ can be computed in $O(M(n)log(n))$ operation in $\mathbb{K}$

- Pade approximant given a series $S = \mathbb{K}[|x|]$ truncated at order $2n$, find $p, q \in \mathbb{K}[x]$ of degree $\leq n$ such that $S = \frac{p}{q} mod x^{2n}$ also can be computed in $O(M(n)log(n))$

8.3 The quasi-Toeplitx case

note that quasi-Vandermonde and quasi-Caucgy cases are more technical and both reduce to the quasi-Toeplitz case.

Three main properties showing that we considers a "good motion" of structured matrix.

(p1) the matrix-vector product can be performed in quasi-optimal time for a quasi-Toeplitz matrix.

(p2) the sum and the product of two quasi-Toeplitz matrices both remain quasi-Toeplitz

(p3) the inverse of an invertible quasi-Toeplitz matrix remains a quasi-Toeplitz matrix.

8.3.1 Matrix-vector product, quasi-Toeplitz case

Proposition 8.1: ΣLU formula

The operator $\phi_+ : A \mapsto A - ZAZ^T$ is invertible. More precisely, the following formula holds

$$A - ZAZ^T = GH^T \quad \text{if and only if} \quad A = \sum_{i=1}^{\aleph} L(x_i)U(y_i)$$

where the x_i's and y_i's are columns of the generator G and H and where, for a column vector $v = [v_0 \dots v_{n-1}]^T$, the matrix $L(v)$ is the lower triangle Toeplitz matrix

$$\begin{pmatrix} v_0 & 0 & \dots & 0 \\ v_1 & v_0 & \dots & 0 \\ \vdots & \ddots & \dots & \vdots \\ v_{n-1} & \dots & v_1 & v_0 \end{pmatrix}$$

and $U(v)$ is the upper triangle Toeplitz matrix $L(v)^T$

This proposition allows us to give an alternative equivalent definition for the displacement rank.

Definition 8.1

The number $\alpha_+(A)$ is the samllest nonnegative integer α s.t. these exists a decomposition of the form

$$A = \sum_{i=1}^{\alpha} L_i U_i$$

where, $L_1, \ldots, L_\alpha$ are lower triangular Toeplitz and Us are upper triangular Toeplitz

If A is given by a concise representation, via a pair of displacement generators (G, H) of dimensions $n \times \alpha$, then the matrix-vector product Av can be performed in $O(\aleph M(n)$ operations in $\mathbb{K}$.

8.3.2 Addition and multiplication in concise representation via generator

Proposition 8.2

Let (T, U) be a displacement generator of A in length α, and Let (G, H) be a displacement generator of B in length β. meanings.... displacement rank of A, B are $\alpha = \alpha(A)$ and $\beta = \alpha(B)$. Then,

- $([T|G], [U|H])$ are displacement generator of $A + B$ with length $\alpha + \beta$
- $([T|G|a], [U|H| - b])$ are displacement generator of AB with length $\alpha + \beta + 1$

where, $V = B^T U, W = ZAZ^T G$ and and where the vector a (resp.b)is the last column of ZA (resp. of ZB^T).

Using a concise representation via displacement generators of length at most α for both A and B , we can compute

- the sum of $A + B$ in $O(\aleph n)$ operations in $\mathbb{K}$
- the product AB in $Mul(n, \alpha) = O(\aleph^2 M(n))$ operations in $\mathbb{K}$

8.3.3 Inversion using the concise representation with generators

Definition 8.2

The displacement operator ϕ_- of a matrix A in $\mathbb{K}^{n\times n}$ is defined by :

$$\phi_-(A) = A - Z^T AZ = A - \text{(A shifted by 1 towards the top and the left)}$$

The corresponding displacement rank α_- is defined as $\alpha_-(A) = rank(\phi_-(A))$.

Lemma 8.0

$\alpha_-(A)$ is the smallest nonnegative integer α such that A can be written $A = \sum_{i=1}^{\alpha} U_i L_i$, where the L_is are lower triangular Toeplitz matrices, and the U_i 's are upper triangular Toeplitz matrices.

ΣUL formula :

$$A_Z^T AZ = \sum_{i=1}^{\alpha} x_i y_i^T \quad \text{if and only if}$$

$$A = \sum_{i=1}^{\alpha} U(rev(x_i))L(rev(y_i))$$

here, for $v = [v_0, \dots, v_{n-1}]^T$ we write $rev(v) = [v_{n-1}, \dots, v_0]^T$

A Toeplitz matrix for ϕ_+ is a Toeplitz matrix for ϕ_-

Proposition 8.1: Conversion $\Sigma LU \leftrightarrow \Sigma UL$

For any matrix A, we have the inequality $|\alpha_+(A) - \alpha_-(A)| \leq 2$. Moreover, conversion can be done in $O(\alpha M(n))$ operations in $\mathbb{K}$.

Theorem 8.4

Let $A \in \mathbb{K}^{n \times n}$ be an invertible quasi-Toeplitz matrix. Then its inverse is also quasi-Toeplitz with $\alpha_+(A^{-1)=\alpha_-(A)}$

Input. A "generic" matrix $A \in \mathbb{K}^{n \times n}$, with $n = 2^k$, represented by generators with respect to the displacement operator ϕ_+.

Output. The inverse A^{-1}, represented by generators with respect to the displacement operator ϕ_-.

1. If $n = 1$, then return A^{-1}.
2. Compute ϕ_+-generators for submatrices $a, b, c, d \in \mathbb{K}^{n/2 \times n/2}$, where $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$.
3. Compute (recursively) ϕ_--generators for $e := a^{-1}$.
4. Compute ϕ_+-generators for $S = d - ceb$.
5. Compute (recursively) ϕ_--generators for $t := S^{-1}$.
6. Return ϕ_--generators of $A^{-1} = \begin{pmatrix} x & y \\ z & t \end{pmatrix}$ via Strassen's formulas $y := -ebt$, $z := -tce$ et $x := e + ebtce$.

8.3.4 Fast solving of quasi-Toeplitz linear systems

Theorem 8.5

Let $A \in \mathbb{K}^{n \times n}$ be an invertible quasi-Toeplitz matrix, with displacement rank α , given by generators G and H of length α . Then, one can compute generators of length α for A^{-1} using $O(\alpha^2 M(n) log(n))$ operations in $\mathbb{K}$.

From this representation of the inverse of A and from a vector $b \in \mathbb{K}^{n\times 1}$, the linear system $Ax = b$ can be solved using an extra $O(\alpha M(n))$ operations in $\mathbb{K}$.

$\vdots$

9 Error Correcting Codes; decoding algorithm