MPRI – Cours 2.12.2

logoINRIA

F. Morain

Lecture II: polynomial arithmetic

2014/09/29

The slides are available on `http://www.lix.polytechnique.fr/Labo/Francois.Morain/MPRI/2013`

Source: von zur Gathen & Gerhard.

I. Fast polynomial multiplication and applications

Def. Let $\mathcal{A}$ be a commutative ring with unity and $\omega \in \mathcal{A}$.
Fourier transform

$$\begin{array}{cccc} \mathcal{F}_\omega : & \mathcal{A}^n & \rightarrow & \mathcal{A}^n \\ & (a_0, a_1, \ldots, a_{n-1}) & \mapsto & (\hat{a}_0, \hat{a}_1, \ldots, \hat{a}_{n-1}) \end{array}$$

$$\hat{a}_i = \sum_{j=0}^{n-1} \omega^{ij} a_j, \quad 0 \le i \le n-1$$

Prop. Suppose ω is a primitive n-th root of unity, i.e., $\omega^n = 1$ and $\omega^i \neq 1$, $1 \le i < n$. Then $\mathcal{F}_\omega$ is 1-1 and

$$\mathcal{F}_\omega^{-1}(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) = \frac{1}{n}\mathcal{F}_{\omega^{-1}}(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}).$$

FFT (cont'd)

Proof: let $a = (a_0, a_1, \ldots, a_{n-1})$. We want

$$\frac{1}{n}\mathcal{F}_{\omega^{-1}} \circ \mathcal{F}_\omega = Id_n.$$

$$\tilde{\alpha}_i = \frac{1}{n}\sum_{k=0}^{n-1} \omega^{-ik}\alpha_k$$

$$n\tilde{\alpha}_i = \sum_k \omega^{-ik} \sum_j \omega^{kj} a_j$$

$$\sum_j a_j \sum_k \omega^{k(j-i)} = \sum_j a_j S_{i,j}.$$

$S_{i,i} = n$ and if $j \neq i$

$$S_{i,j} = \sum_{k=0}^{n-1} (\omega^{j-i})^k = \frac{1-(\omega^{j-i})^n}{1-\omega^{j-i}} = 0.$$

$\Rightarrow \tilde{\alpha}_j = a_j$. $\square$

Application: multiplication of polynomials

$A = \sum a_i X^i, \quad B = \sum b_i X^i, \quad P = AB = \sum p_i X^i$

$$a = (a_0, a_1, \ldots, a_{n-1}, \underbrace{0, 0, \ldots, 0}_{n \text{ coefficients}}),$$

$$b = (b_0, b_1, \ldots, b_{n-1}, \underbrace{0, 0, \ldots, 0}_{n \text{ coefficients}}).$$

Let ω be a primitive $2n$-th root of 1.

$$\mathcal{F}_\omega(a) = (A(\omega^0), A(\omega^1), \ldots, A(\omega^{2n-1}))$$

$$\mathcal{F}_\omega(b) = (B(\omega^0), B(\omega^1), \ldots, B(\omega^{2n-1}))$$

$$\mathcal{F}_\omega(p) = (P(\omega^0), P(\omega^1), \ldots, P(\omega^{2n-1}))$$

Term-wise product:

$$\mathcal{F}_\omega(a) \otimes \mathcal{F}_\omega(b) = (\hat{a}_0\hat{b}_0, \hat{a}_1\hat{b}_1, \ldots, \hat{a}_{2n-1}\hat{b}_{2n-1}) = \mathcal{F}_\omega(p).$$

To compute $A(X)B(X)$:
1. compute $\mathcal{F}_\omega(a)$, $\mathcal{F}_\omega(b)$;
2. compute $p = \mathcal{F}_\omega(a) \otimes \mathcal{F}_\omega(b)$;
3. recover $P = (p_0, p_1, \ldots, p_{2n-1}) = \mathcal{F}_\omega^{-1}(p)$.

Fast evaluation

Pb. evaluate
$$\hat{x}_k = \sum_{m=0}^{N-1} x_m \omega^{mk}, 0 \le k \le N-1.$$

Naive solution: N^2 multiplications.

Better: assume $N = N_1 N_2$. Rewrite

$$\begin{aligned} m &= N_1 m_2 + m_1, \\ k &= N_2 k_1 + k_2, \end{aligned}$$

with $0 \le m_1, k_1 \le N_1 - 1$ and $0 \le m_2, k_2 \le N_2 - 1$.

$$\hat{x}_k = \sum_{m_1=0}^{N_1-1} \omega^{N_2 m_1 k_1} \omega^{m_1 k_2} \sum_{m_2=0}^{N_2-1} x_{N_1 m_2 + m_1} \omega^{N_1 m_2 k_2}.$$

Fast evaluation (cont'd)

Write
$$\hat{x}_k = \sum_{m_1=0}^{N_1-1} \omega_1^{m_1 k_1} \omega^{m_1 k_2} \sum_{m_2=0}^{N_2-1} x_{N_1 m_2 + m_1} \omega_2^{m_2 k_2}$$
with $\omega_1 = \omega^{N_2}$ et $\omega_2 = \omega^{N_1}$.

Key remark: ω_u is a primitive N_u-th root of 1.

i.e., compute N_1 DFT of length N_2, followed by multiplications by $\omega^{m_1 k_2}$, followed by N_2 DFT of length N_1.

Cost:

$$N_1(N_2^2) + N_1 N_2 + N_2(N_1^2) = N_1 N_2 (N_1 + N_2 + 1) < N_1 N_2)^2$$

The case $N = 2^t$

Special case $N_1 = 2$, $N_2 = 2^{t-1}$. Then

$$\hat{x}_k = \sum_{m=0}^{N/2-1} x_{2m} (\omega^2)^{mk} + \omega^k \sum_{m=0}^{N/2-1} x_{2m+1} (\omega^2)^{mk}.$$

Since $\omega^{N/2} = -1$

$$\hat{x}_{k+N/2} = \sum_{m=0}^{N/2-1} x_{2m} (\omega^2)^{mk} - \omega^k \sum_{m=0}^{N/2-1} x_{2m+1} (\omega^2)^{mk}.$$

Cost:
$$F(N) = 2F(N/2) + N/2$$
or
$$\frac{F(N)}{N} = \frac{F(N/2)}{N/2} + 1/2 = F(1) + t/2 = t/2$$

which is $F(N) = \frac{1}{2} N \log_2 N$.

FFT: Maple code

```
# x[0..N-1], N is a power of 2, W^(2^N) = 1.
FFTp := proc(p, omega, x, N)
local k, m, Y1, Y2, X, xx, omegak;
  if N = 2 then
    X[0]:=x[0]+x[1] mod p;
    X[1]:=x[0]-x[1] mod p;
    RETURN(X);
  else
    for m from 0 to N/2-1 do xx[m]:=x[2*m]; od;
    Y1:=FFTp(p, omega^2, xx, N/2);
    for m from 0 to N/2-1 do xx[m]:=x[2*m+1]; od;
    Y2:=FFTp(p, omega^2, xx, N/2);
    for k from 0 to N/2-1 do
      omegak:=omega^k mod p; # do better
      X[k]     :=Y1[k] + omegak*Y2[k] mod p;
      X[k+N/2]:=Y1[k] - omegak*Y2[k] mod p; # reuse
    od;
    RETURN(X);
  fi;
end:
```

FFT: complementary remarks

Squaring: $A(X)^2$
1. compute $\mathcal{F}_\omega(a)$;
2. compute $p = \mathcal{F}_\omega(a) \otimes \mathcal{F}_\omega(a)$ (hence a lot of squares);
3. recover $P = (p_0, p_1, \ldots, p_{2n-1}) = \mathcal{F}_\omega^{-1}(p)$.

Reusing FFT's: if a lot of multiplications by B, cache $\mathcal{F}_\omega(B)$. Typical use in fast exponentiation, division.

Fürer: better complexity; perhaps useful.

Implementations:
- see Shoup95, etc. Take care to the existence of primitive roots of unity (CRT for $\mathbb{Z}[X]$).
- GMP, MPIR, FLINT.

Fast division

$\deg(M(X)) = n$, $\deg(P(X)) < 2n$.

Prop. Write $P = QM + R$ with $\deg R < \deg M$. Let $I(X) = X^{2n} \div M$; $g = P \div X^n$. Then $Q = (g(X)I(X)) \div X^n$.

Proof: $H(X) = (g(X)I(X)) \div X^n$. We get

$$X^{2n} = M(X)I(X) + \mu(X), \deg \mu < n,$$

$$P(X) = X^n g(X) + \rho(X), \deg \rho < n.$$

Write

$$X^n(P(X) - H(X)M(X)) = X^n(P(X) - X^n g(X)) - g(X)(M(X)I(X) - X^{2n})$$
$$-M(X)(H(X)X^n - g(X)I(X))$$

all polynomials of the rhs have degree $< 2n$, hence $P(X) - H(X)M(X)$ has degree $< n$, which implies $H = Q$. $\square$

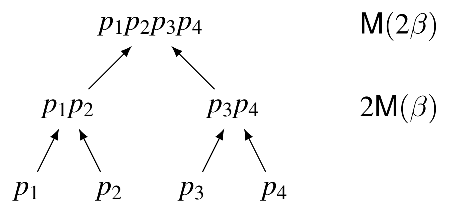
Typical use: computing $X^e \bmod (MX, p)$

```
# e = array of t+1 bits
PolXpowerMod:= proc(p, MX, X, e, t)
local b, i, IX;
    IX:= Quo(X^(2*degree(MX, X)), MX, X) mod p;
    b:= X;
    for i from t-1 by -1 to 0 do
      b:= PolSquare(p, b, MX, X);
      b:= PolModFast(p, b, MX, IX, X);
      if e[i] = 1 then
        b:= Rem(b * X, MX, X) mod p;
      fi
    od;
    RETURN(b);
end:
```

Rem. $I(X)$ is fixed through the loop, therefore we can cache its FFT.

II. Product trees: principles and applications

Imagine all p_i's have the same size β.



Product tree: $2\mathsf{M}(\beta) + \mathsf{M}(2\beta)$.

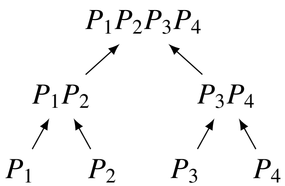
Naive case: $\underbrace{p_1p_2}_{\mathsf{M}(\beta)} + \underbrace{(p_1p_2)p_3}_{\mathsf{M}(2\beta,\beta)} + \underbrace{(p_1p_2p_3)p_4}_{\mathsf{M}(3\beta,\beta)} \approx 6\mathsf{M}(\beta)$.

Comparison: $4\mathsf{M}(\beta)$ vs. $\mathsf{M}(2\beta)$? Equal if $\mathsf{M}(\beta) = \beta^2$, product tree better if $\mathsf{M}(\beta) = \beta^a$, $a < 2$.

General principle: only the last step counts.

With polynomials

Goal: compute $Z = P_1 \cdots P_m$ for polynomials $P_i(X)$.

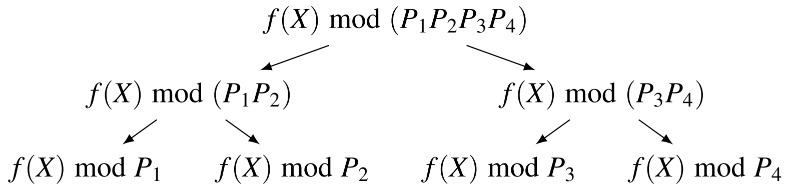


Typical application: PolyFromRoots, i.e., given $(x_i)_{0 \leq i < n}$, build $P(X) = \prod(X - x_i)$. Cost is $O(\mathsf{M}(n) \log n)$.

Fast multipoint evaluation

Goal: compute $f(X) \bmod P_i(X) = X - x_i$ for all i.

Use a remainder tree, i.e.,



Key property: $f(X) \bmod (X - x_i) = f(x_i)$

Complexity: $O(\mathsf{M}(n) \log n)$.

Fast resultant and discriminant

$$P(X) = \prod_{i=0}^{n-1}(X - \alpha_i), Q(X) = \prod_{j=0}^{m-1}(X - \beta_j)$$

$$\mathrm{Res}(P, Q) = \prod_{i,j}(\alpha_i - \beta_j) = \prod_{i=0}^{n-1} Q(\alpha_i).$$

Algorithm: use fast multipoint evaluation to compute all $Q(\alpha_i)$; finish with a product tree.

$$\begin{aligned}\mathrm{Disc}(P) &= (-1)^{n(n-1)/2}\mathrm{Res}(P, P') \\ &= \prod_{i<j}(\alpha_i - \alpha_j)^2 \\ &= (-1)^{n(n-1)/2}\prod_{i=0}^{n-1} P'(\alpha_i).\end{aligned}$$

Rem. The resultant can be computed using Euclid's algorithm when the roots are not known.

III. Factoring polynomials over finite fields

Goal: Given $f(X) \in \mathbb{F}_q[X]$ of degree n, write

$$f(X) = f_1(X)^{e_1} f_2(X)^{e_2} \ldots f_k(X)^{e_k}$$

where $f_i \neq f_j$ is irreducible.

Generic approach:

1. Squarefree factorization (SQF): compute $f_i(X)$ s.t.

$$f = g_1^1 g_2^2 \cdots g_m^m$$

   with $\gcd(g_i, g_j) = 1$, g_i squarefree;
2. Distinct degree factorization (DDF): for all $h \in \{g_1, g_2, \ldots, g_m\}$, look for the degrees of $f_i \mid h$ using

$$\gcd(X^{q^d} - X, h(X))$$

   for $d \leq \deg(h)$;
3. Equal degree factorization (EDF): find all degree d factors of h.

A) Squarefree factorization

$$f = \sum_{i=0}^{n} a_i X^i \mapsto f' = \sum_{i=1}^{n} i a_i X^{i-1}, \quad a_i \in \mathbb{F}$$

Def. If $f = \prod_{i=1}^{r} f_i^{e_i}$ where f_i is irreducible, then the squarefree part of f is $sqfp(f) = \prod_i f_i$.

Key remark: if $f = g^2 h$, then $f' = g(2g'h + gh')$ and $g \mid u = \gcd(f, f')$. If $char(\mathbb{F}) = 0$, $f' \neq 0$, $\deg(f') < n$ and u a proper divisor of f.

Prop. If $char(\mathbb{F}) = 0$, then $u = \prod_i f_i^{e_i - 1}$ and $sqfp(f) = f/u$.
Proof: All summmands in

$$(*) \quad f' = \sum_{i=1}^{r} e_i \frac{f}{f_i} f_i'.$$

are divisible by $f_i^{e_i}$ except $e_i \frac{f}{f_i} f_i'$ which is divisible *a priori* by $f_i^{e_i-1}$. If $char(\mathbb{F}) = 0$, then $f_i' \neq 0$ and $e_i \neq 0$. $\square$

Squarefree decomposition

Def. squarefree decomposition: $f = g_1 g_2^2 \cdots g_m^m$, g_i monic squarefree coprime; g_i is the product of polynomials dividing f exactly i times.
Prop. $sqfp(f) = g_1 g_2 \cdots g_m$.
Lemma. If $g = g_1 g_2 \cdots g_m$ and $h = \sum_{i=1}^{m} c_i g_i' g/g_i$ for some $c_i \in \mathbb{F}$, then for all $c \in \mathbb{F}$

$$\gcd(g, h - cg') = \prod_{c_j = c} g_j.$$

Proof:

$$g' = \sum_{i=1}^{m} g_i' \frac{g}{g_i} \Rightarrow h - cg' = \sum_{i=1}^{m} (c_i - c) g_i' \frac{g}{g_i}.$$

For $i \neq j$, g_j divides each summand, $\gcd(g_j, g_j') = \gcd(g_j, g/g_j) = 1$.

$$\gcd(g_j, h - cg') = \gcd(g_j, (c_j - c) g_j' \frac{g}{g_j}) = \gcd(g_j, c_j - c). \quad \square$$

Yun's algorithm

1. $u := \gcd(f, f')$; $v_1 := f/u$; $w_1 := f'/u$.
2. $i := 1$;

repeat
  $h_i := \gcd(v_i, w_i - v_i')$; $v_{i+1} := v_i/h_i$; $w_{i+1} := (w_i - v_i')/h_i$;
  $i := i + 1$;
until $v_i = 1$;
RETURN $((h_1, 1), (h_2, 2), \ldots, (h_{i-1}, i-1))$ (some h_i can be 1).

Justification: when $char(\mathbb{F}) = 0$

$$u = g_2 g_3^2 \cdots g_m^{m-1} = \prod_i f_i^{e_i - 1}, v_1 = \prod_i f_i = \prod_j g_j, \quad g_j = \prod_{e_i = j} f_i,$$

$$w_1 = \frac{f'}{u} = \frac{1}{u} \sum_i e_i f_i' \frac{f}{f_i} = \sum_i e_i f_i' \frac{v_1}{f_i} = \sum_{0 < j \leq m} j g_j' \frac{v_1}{g_j}.$$

$$h_i = g_i, \quad v_{i+1} = \prod_{i < j \leq m} g_j, \quad w_{i+1} = \sum_{i < j \leq m} (j - i) g_j' \frac{v_{i+1}}{g_j}.$$

Cost: $O(\mathsf{M}(n) \log n)$.

The case of finite fields (1/2)

Prop. $\forall f \in \mathbb{F}_q[X], f' = 0 \iff f = g^p$ in $\mathbb{F}_q[X]$.

SquarefreePart(f)

1st case: $f' = 0$; write $f = g^p$ and return $sqfp(g)$.

2nd case: $f = g^p h$ where $h \in \mathbb{F}[X]$, $h' \neq 0$.
In other words, $h = \prod_{p \nmid e_i} f_i^{e_i}$.

$f' = g^p h'$ and $u = \gcd(f, f') = g^p \gcd(h, h') = g^p \prod_{p \nmid e_i} f_i^{e_i - 1}$.
$v = f/u = h/\gcd(h, h') = \prod_{p \nmid e_i} f_i$
$w = \gcd(u, v^n) = \gcd(g^p \prod_{p \nmid e_i} f_i^{e_i - 1}, \prod_{p \nmid e_i} f_i^n) = \prod_{p \nmid e_i} f_i^{e_i - 1}$.
$u/w = \prod_{p \mid e_i} f_i^{e_i} = (\prod_{p \mid e_i} f_i^{e_i/p})^p = F_2^p$.

Cost: $O(\mathsf{M}(n) \log n)$ operations in $\mathbb{F}_q$.

Recursively: $sqfp(f) = v \times sqfp(F_2)$.

Total cost: $O(\mathsf{M}(n) \log n + n \log(q/p))$.

Numerical example

$f = ab^2c^2d^6e^8$ in $\mathbb{F}_2[X]$ for irreducible a, b, c, d, e.

$f' = a'(b^2c^2d^6e^8)$

$u_1 = \gcd(f, f') = (1)(b^2c^2d^6e^8), v_1 = a,$

$w_1 = \gcd(u_1, v_1^n) = 1, u_1/w_1 = (bcd^3e^4)^2 = F_2^2;$

$u_2 = \gcd(F_2, F_2') = (d^2)(e^4), v_2 = bcd, w_2 = d^2,$
$u_2/w_2 = (e^2)^2 = F_3^2;$

$u_3 = \gcd(F_3, F_3') = e^2, v_3 = 1, w_3 = 1, u_3/w_3 = (e)^2 = F_4^2;$

$u_4 = 1, v_4 = e, w_4 = 1, u_4/w_4 = 1.$

$sqfp(f) = v_1v_2v_3v_4 = (a)(bcd)(1)(e).$

Squarefree decomposition (1/2)

When $p = char(\mathbb{F})$ and $m < p$, Yun's algorithm gives the right answer.

Prop. Yun's algorithm computes $h_i = \prod_{j \equiv i \bmod p} g_j$ for $1 \leq i < p$ and $h_i = 1$ for $i \geq p$. (Note that some h_i can also be $= 1$.)

Coro. Yun's algorithm is enough for $m < p$.

Proof: $f' = g^p h'$, $h = \prod_{p \nmid e_i} f_i^{e_i}$;
$u = \gcd(f, f') = g^p(\prod_{p \nmid e_i} f_i^{e_i - 1})$, $v_1 = f/u = \prod_{p \nmid e_i} f_i$,

$$w_1 = f'/u = \frac{h'}{\prod_{p \nmid e_i} f_i^{e_i-1}} = \frac{1}{\prod_{p \nmid e_i} f_i^{e_i-1}} \sum_{p \nmid e_i} e_i f_i' \frac{h}{f_i} = \sum_{p \nmid e_i} e_i f_i' \frac{v_1}{f_i}.$$

Using the Lemma

$$h_1 = \gcd(v_1, w_1 - v_1') = \prod_{e_i = 1 \bmod p} f_i, \quad v_2 = v_1/h_1 = \prod_{e_i \bmod p \notin \{0,1\}} f_i.$$

Squarefree decomposition (2/2)

Key remark: $f h_1^{-1} h_2^{-2} \cdots h_{p-1}^{-p+1} = z^p$. (*Proof:* $h_j = \prod_{e_i \equiv j \bmod p} f_i$.)

Refining: we have to split the h_i's depending on e_j and not only on $e_j \equiv i \bmod p$.

Suppose we have h_i and $\sigma = SQF(z) = ((z_1, \varepsilon_1), \ldots, (z_n, \varepsilon_n))$.

If $f_j \mid h_i$ and $j > p$, then f_j divides exactly one factor in σ.

$S := \emptyset$
for $i := 1$ **to** $p - 1$ s.t. $h_i \neq 1$ **do**
  for $j := 1$ **to** n **do**
    $g := \gcd(h_i, z_j)$
    if $g \neq 1$ **then**
      $h_i := h_i/g$; $z_j := z_j/g$;
      $S := S \cup (g, i + p\varepsilon_j)$;
      if $h_i = 1$ **then** break;
  if $h_i \neq 1$ **then** $S := S \cup (h_i, i)$;
$S := S \cup (z_j, p \times \varepsilon_j)$ for the $z_j \neq 1$.
return sort(S).

B) Distinct degree factorization (DDF)

Input: squarefree monic $f(X)$ of degree $n > 0$.
Output: $f(X) = (g_1, g_2, \ldots, g_s)$ of f.
 $h_0 := X; f_0 := f; i := 0;$
 repeat
  $i := i + 1;$
  $h_i := h_{i-1}^q \bmod f_{i-1};$
  $g_i := \gcd(h_i - X, f_{i-1}), \quad f_i := f_{i-1}/g_i;$
 until $f_i = 1$;
 RETURN $(g_1, g_2, \ldots, g_s)$.

Analysis: $O(s\mathsf{M}(n)\log(nq))$ operations in $\mathbb{F}_q$.

Rem. stop as soon as $\deg(f_i) < 2(i + 1)$.

Rem. use the artillery presented before to compute $h_{i-1}^q \bmod f_{i-1}$.

C) Equal degree factorization (EDF): q odd

Hyp. $f = f_1 f_2 \cdots f_r$ with f_i irreducible of degree d, $f_i \neq f_j$.

$$R = \mathbb{F}_q[X]/(f) \simeq \mathbb{F}_q[X]/(f_1) \times \mathbb{F}_q[X]/(f_2) \times \cdots \mathbb{F}_q[X]/(f_r)$$

$$= R_1 \times R_2 \times \cdots \times R_r, \text{ with each } R_i \simeq \mathbb{F}_{q^d}.$$

Idea: For $a \in \mathbb{F}_q[X]$, put $\chi(a) = (\chi_1(a), \ldots, \chi_r(a))$ with $\chi_i(a) = a \bmod f_i$. One has $f_i \mid a$ iff $\chi_i(a) = 0$. If not all of the $\chi_i(a)$ are non-zero, then $\gcd(a, f)$ will be non-trivial.

Lem. S group of non-zero squares in $\mathbb{F}_q$, is a multiplicative subgroup of $\mathbb{F}_q$ of order $(q-1)/2$; $a \in S$ iff $a^{(q-1)/2} = 1$.

Algorithm: compute $b = a^{(q-1)/2} \bmod f$ for random a and compute $\gcd(b-1, f)$. It will split f with probability $\geq 1 - (1/2)^{r-1}$.

Analysis: to find r factors, needs $O((\log r)(d \log q + \log n)\mathsf{M}(n))$.

Equal degree factorization (EDF): q even

$q = 2^k$; all f_i are irreducible of degree d.
Let

$$T_m(X) = X^{2^{m-1}} + X^{2^{m-2}} + \cdots + X^4 + X^2 + X.$$

(a) T_m is $\mathbb{F}_2$-linear and $T_m(\alpha) \in \mathbb{F}_2$ for all α.
(b) $X^{2^m} - X = T_m(X)(T_m(X) - 1)$. Hence $T_m(\alpha) = 0$ or 1 with probability $1/2$.
(c) $\chi_i(T_{kd}(\alpha)) \in \mathbb{F}_2$ for all $\alpha \in R$. Hence $T_{kd}(\alpha) \in \mathbb{F}_2$ with proba $2^{1-r} \leq 1/2$.

Algorithm: compute $b = T_{kd}(a) \bmod f$ for random a until $\gcd(b, f)$ splits f.

Factoring polynomials over finite fields

- a problem rather well understood; many more algorithms available (Berlekamp, Shoup's improvements, von zur Gathen, etc.);
- if field is small, brute force is possible;
- still some work to be done for large p;
- useful for computing discrete logarithms.