Factoring Polynomials

Lecture 15

Why do we want to factor a polynomial?

- NO if we want to find approximate roots of a univariate polynomial. Use a numerical method.
- YES to simplify a result which may appear smaller when factored. iffy...
- $x^{12}-1 = (x-1)¢(x+1)¢(x^2+1)¢(x^2-x+1)¢(x^2+x+1)¢(x^4-x^2+1)$
- YES to simplify MULTIVARIATE root-finding.
- YES to do (traditional version) partial fraction expansion for integration.
- YES, applications in coding theory/ error correcting codes (factor over GF[2]) and computational number theory.

We want to avoid really factoring over Z[x]

- Decide if this is really a misstated request for zero-finding.
- Attempt cheap proofs of irreducibility.
- Attempt cheap special recognition.
- Attempt cheap square-free factorization.
- Attempt (relatively) cheap distinct-degree factorization.
- Attempt to grow mod q factors via Hensel lifting to factorization over the integers.
- Factoring integers in Z is nominally a subset of this problem, but really uses different technology, has different objectives.

Zerofinding problem $\neq$ Factoring

- Does the user expect all linear factors for a polynomial in one variable? (Or linear + quadratic conjugate pair factors)?
- Are coefficients representable in floating point?
  - If so, redirect to Conventional Numerical methods
- If not representable in floating point, consider
  - Exact rational root isolation methods “Sturm Sequences” or similar
  - Extended “bigfloat” zerofinding
- Does the user wish only real zeros, guaranteed isolated? Proceed directly to Sturm Sequences, or Bisection, or Descartes Rule of Signs, and/or high precision floats.

A random polynomial is nearly always irreducible

- (Knuth, Art of Comp. Progr. vol II, ex 4.6.2)
- But the interesting cases are in that small set of polynomials which actually factor.

- Actually, Knuth's work is fairly thorough background on this material, though VzG may have more recent material.

Irreducibility tests can help

- Eisenstein irred. criterion: polynomial f(x): if all the coefficients (except possibly the first) are divisible by a prime p, and the constant coefficient is *not* divisible by p^2, then f(x) is irreducible. Various transformations of the polynomial can also help) http://www.mathpages.com/home/kmath406.htm
- If monic f(x) mod p is irreducible mod p, then so is f.
- (the reverse is not true: x^4+1 always factors mod p but not over the integers.)
- If p is a prime number, $x^{p-1}+x^{p-2}+..+1$ is irreducible (Gauss)

More Irreducibility tests

- Ore's criterion (based on Newton Polygon, [Zippel 19.1])
- Evaluate $a_1=f(c_1)$, $a_2=f(c_2)$, $a_3=f(c_3)$. If they are all prime and f is monic, square-free, we can deduce some restrictions on g,h where f=g¢ h; perhaps deduce irreducibility.
- Probabilistic primality testing of univariate polys (Weinberger).
- If f factors into incommensurate factor degrees in different finite fields, e.g. If deg(f)=4 and factors mod two primes are of degree 2,2 and 1,3, then it is irreducible (basis for factoring, anyway...)

Hilbert's Irreducibility Thm

- (1892) For an irreducible polynomial f 2 Q[x,y], the univariate polynomial f(x,a) 2 Q[x] is irreducible for most a 2 Z. (Helpful especially in reducing from more than 2 variables to just 2. E. Kaltofen used this to find a probabilistic polynomial time multivariate factoring procedure.)
- For additional characterizations, as well as a substantial bibliography, see von zur Gathen, 14.9 *et seq*. Note that progress on many of the open problems suggested there are unlikely to affect any practice of computing, but may serve to sharpen complexity analysis.

Often the polynomials (and their factors) are well known

- Is it a cyclotomic polynomial?
- $\prod_{1\cdot\ k\ \cdot\ n,\ gcd(k,n)=1} (x-\omega^k)$
  - various systematic ways of generating factors over the integers and Gaussian integers
- Was it produced by multiplying stuff together recently (memoization)

Removing "content"

- $9x^2-9$ factors into $9¢ (x^2-1)$ by removing the gcd of the coefficients. Whether to factor 9 now (or ever) depends on whether you want to factor the polynomial content in Z. Factoring potentially large integers is "harder than factoring large polynomials" in some sense.
- This helps with multivariate factors too:
- $-y^4+x^2y^2+y^2-x^2=(x^2-y^2)¢(y^2-1)$

Square-free factorization

- $f(x)=f_1(x)f_2(x)^2f_3(x)^3\ldots f_k(x)^k$
- Observe that if $f=g^n$¢ h and g,h depend on x:
- $df/dx = f' = g^n$¢ $h'+ng^{n-1}$¢ g'¢ $h = g^{n-1}$¢$(g$¢ $h'+n$¢ g'¢ $h)$
- so g^{n-1} divides $r=\gcd(f,f')$ (not equal to gcd…)
- Repeat to try to factor r.
- A slightly better sequence is to compute $\gcd(f/r, f-f')$. (D. Yun), still reducing multiplicity by one each time.
- Iterate over all variables in f… ultimately we get f_1¢ f_2¢ f_3¢ …¢ f_k

Distinct Degree Factorization helps too:

- We have, from square-free factorization, partially decomposed f. Now look at each

$f_i(x)= f_{i,1}(x)f_{i,2}(x) \ldots f_{i,r}(x)$ where $f_{i,j}$ is a product of all the irreducible factors of f_i of degree j. *Factoring the $f_{i,j}$ is the hardest part and is done via finite field factorizations and lifting.*

How to do Distinct Degree Factorization

- Only practical over finite fields, univariate.
- Let $f(x) = f_1 f_2 \ldots f_k$ with f_j product of irreducible polyns of degree j, and be square free monic over F_q of degree n where $q=p^r$.
- Fermat's little theorem says that each element of F_q is a zero of x^q-x, i.e.
- $\prod_{\alpha\ 2\ F/q}(x-\alpha) = x^q-x$.
- Since f is square free, f_1 is the gcd(f,x^q-x) and the product of all the monic polys of degree less than r is $x^{(q\hat{}r)}$-x. so we compute f_r as gcd(f, $x^{(q\hat{}r)}$-x)
- (There is a trick here; we compute large values of x^q^r by repeated squaring modulo f(x). Another trick: remove factors as fast as you can find them.)

What’s left?

- Factoring a univariate polynomial all of whose factors are of the same degree.
- Reducing multivariate factoring over the integers to univariate factoring over finite fields
- Relate factoring over FINITE fields to factoring “over the rationals” (which reduces to “over the integers”).

Factoring over finite fields does not immediately tell us about rational factors

- several factors over several finite fields of the squarefree… (X+1)¢(X^2+1)¢(X^3+1):
- Mod   factors
- 3   (X+1)^4¢(X^2+1)
- 7   (X-3)¢ (X+1)^2¢(X+2)¢(X^2+1)
- 11   (X+1)^2¢(X^2+1)¢(X^2-X+1)
- But none of these are square-free!

Particularly fiendish problems are of this form

- $\prod$(x§ sqrt(2)§ sqrt(3) § ...sqrt(p_k)) known as Swinnerton-Dyer polynomials, which are irreducible but factor in (most) finite fields. (p_k is k^{th} prime number)
- x^4+1 factors in EVERY finite field but not over integers (Knuth prob. 4.6.2.12,13)
- Why not use CRA? We would still have to piece together different factorizations; we are more successful using Hensel lifting.

If we factor in a finite field we may have to overcome several pieces of misinformation

- Wrong degrees: degree 1 + degree 2 ➔ degree 3 polynomial in answer, perhaps.
- Wrong coefficients: use a bound on the coefficients in the factors to limit growth via Hensel lifting.

g(x) mod p, mod p^2, mod p^4 etc until p^n exceeds some coefficient bound, e.g. Mignotte's bound:

Suppose g¢ h divides f, deg(h)=k. Then $||h||_1 \cdot 2^k||f||_2 \cdot 2^k||f||_1$. (other such crude norms can be found..)

…1 norm is max of coeffs, 2-norm is sqrt of sum of squares, 1-norm is sum of abs vals.

Consider special case of product of linear factors

If h is a product of linear factors and x^q-x is a product of all linear factors, gcd(h,x^q-x) = h. No help. But

$x^q-x=x¢(x^{(q-1)/2}-1)¢(x^{(q-1)/2}+1) = x¢\ r\ ¢\ s.$

Computing gcd(h,s) may split h, since some of the factors of h will be in r, some in s. This actually splits h into classes of factors which are quadratic residues or not.

What if gcd(h,s)=h (i.e. no splitting?)

Try to split, again.

If $h(x)$ doesn't split, try $h(x-b)$ with $w(x)=x^q-x$.

or alternatively, $\gcd(h(x), w(x-b))$. Try for a bunch of random values of b. How likely is this to find a factor? Probably. (Fewer than 2 tries on average should be needed).

Generalize to factors of higher order?

- Idea is to find a set of polynomials comparable to w(x) such that gcd(h,w) splits out factors of higher degree. Probably. The construction and analysis is in (for example) Zippel's text.
- This (Cantor-Zassenhaus method) looks neat. Is it used? Apparently. Berlekamp method may be faster.

Still a contender, esp. mod 2: variants of algorithms by E. Berlekamp

- Large prime/ small prime versions (c. 1968-1970)
- Based on linear algebra
- Provides a strong tool, in combination with the previous material to factor multivariate polynomials over the integers.
- Numerous "improvements" some of which may be faster in particular regions of the problem domain, but may not. (vzG ch. 14) (Possible project: find / implement really fast versions, benchmarks.)

Berlekamp Factoring Algorithm: Goal

- We wish to factor univariate monic polynomial f over a small finite field of order *q*. Let deg(*f*(*x*))=n. The key idea is to find and exploit solutions, *g*(*x*), of the congruence

$$g(x)^q - g(x) = 0 \mod f(x).$$

  Because *q* is the order of the finite field, it is not hard to show that the coefficients of *g* satisfy a system of *n* linear equations. ..

Berlekamp Factoring Algorithm: Outline

$$(Q - I)\,\mathbf{g} = \mathbf{0}.$$

Here Q and I are n £ n matrices over F_q. The entries of Q are computed from the polynomial f(x). One then finds solution vectors, **g**, and corresponding polynomials, g. We use the fact that

$$g(x)^q - g(x) = \prod (g(x) - s),$$

where s runs over all q elements in the field. Since we now have a factorization of a multiple of $f(x)$, we can factor $f(x)$ by computing its gcd with each factor of the multiple.

- *"Factoring Polynomials over Large Finite Fields", Mathematics of Computation* **24**:713-735 (1972);

Berlekamp mod-p factoring, details (Knuth vol 2)

- $u(x)$ coefficients in $\{0,1,\ldots,p-1\}$ degree n.
- remove multiple factors by $d=\gcd(u,u')$.
- If $d=1$ then u is squarefree.
- (If $d=u$, $u'=0$ hence $u(x)=v(x^p) = v(x)^p$)
- This previous line is an important identity:
- $(v_1(x)+v_2(x))^p = v_1(x)^p + \text{binom}(p,1)¢\ v_1(x)^{p-1}¢\ u_2(x) + \ldots +v_2(x)^p$ where all binom coeffs are divisible by p and therefore 0, so $(v_1(x)+v_2(x))^p = v_1(x)^p + v_2(x)^p$; $v(x)^p = v(x^p)$, also $a^p = a \bmod p$ for constants a in Z_p.

Consider factoring u=$f_1(x)$ ¢ ... ¢ $f_r(x)$

- $f_1, \ldots, f_r$ are relatively prime, so for a set of integers $\{s_1, \ldots, s_r\}$ there is a unique v(x) such that

$v(x)$ ´ s_1 (modulo f_1) i.e. s_1 is remainder after dividing v(x) by f_1 mod p

...

$v(x)$ ´ s_r (modulo f_r)

also $\deg(v) < \deg(f_1) + \ldots + \deg(f_r) = \deg(u)$

(By Chinese Remainder Thm.)

The polynomial v(x) gives us a way to get at factors of u(x)

- if r ¸ 2 and $s_1 \neq s_2$ then gcd(u(x),v(x)-s_1) will be divisible by $f_1(x)$ but not by $f_2(x)$. That means if we can find appropriate solutions v(x), we can get information on the factors of u.
- Observe:
- $v(x)^p$ ´ $s_j^p = s_j$ ´ v(x) mod $f_j(x)$ for 1 · j · r therefore
- $v(x)^p$ ´ v(x) modulo u(x), deg(v) < deg(u) [*]

The relationship of u and v

- Also x^p-x ´ (x-0)¢ (x-1) ¢ ... ¢ (x-(p-1)) modulo p
- and
- v(x)^p –v(x) = (v(x)-0) ¢ .... ¢ (v(x)-(p-1)) [**] is an identity for any poly v(x), when we are working mod p.
- If v(x) satisfies [*]
- v(x)^p ´ v(x) modulo u(x), deg(v) ‹ deg(u) [*]
- then u(x) divides the lhs of [**] so every irreducible factor of u(x) must divide one of the p relatively prime factor os the rhs of [**]. That is, <u>all</u> solutions of [*] must have the form of v(x) for sol {s_1, ...,s_r}, so there are exactly p^r solutions of [*].

solving the congruences for v

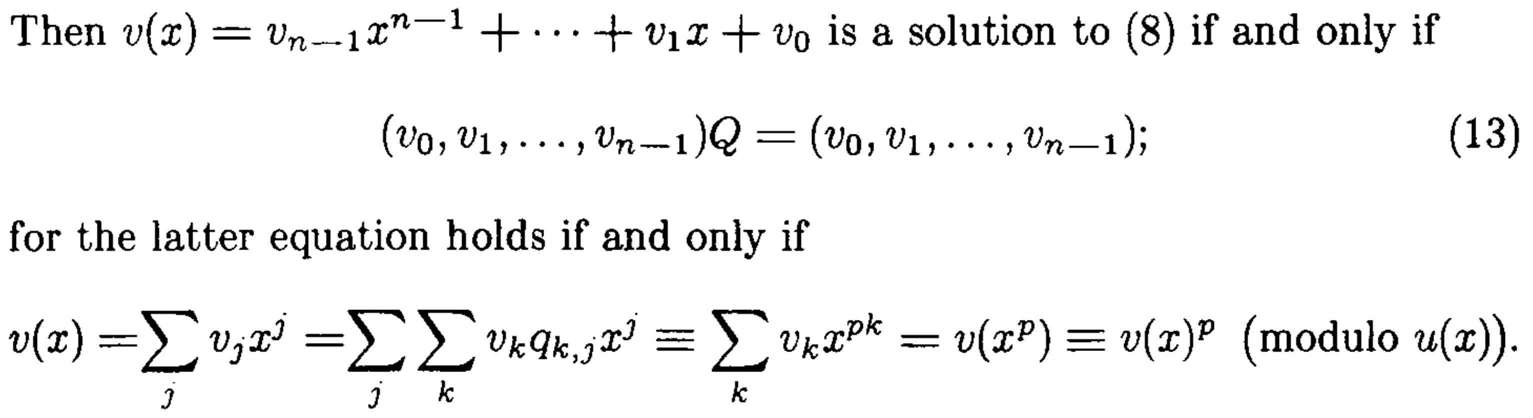
- let deg(u)=n

$$Q = \begin{pmatrix} q_{0,0} & q_{0,1} & \cdots & q_{0,n-1} \\ \vdots & \vdots & & \vdots \\ q_{n-1,0} & q_{n-1,1} & \cdots & q_{n-1,n-1} \end{pmatrix}$$

where

$$x^{pk} \equiv q_{k,n-1}x^{n-1} + \cdots + q_{k,1}x + q_{k,0} \quad (\text{modulo } u(x)).$$

solving the congruences for v



Then $v(x) = v_{n-1}x^{n-1} + \cdots + v_1 x + v_0$ is a solution to (8) if and only if

$$(v_0, v_1, \ldots, v_{n-1})Q = (v_0, v_1, \ldots, v_{n-1}); \tag{13}$$

for the latter equation holds if and only if

$$v(x) = \sum_j v_j x^j = \sum_j \sum_k v_k q_{k,j} x^j \equiv \sum_k v_k x^{pk} = v(x^p) \equiv v(x)^p \pmod{u(x)}.$$

these relations form the basis for Berlekamp's algorithm (figures from Knuth vol 2)

Lenstra-Lenstra-Lovasz (L^3) Lattice Reduction

- Let α be an approximation of some real zero of u(x). The minimal polynomial for α is an irreducible polynomial v(x) that divides u(x). Repeat this process with u/v.
- How to find v? First search for linear, then quadratic, etc.
- Approach to find a degree k factor:
- create a k+1 dimensional lattice L_k that has a basis of:

Basis:

- $(\alpha^k,0,\ldots,0),(0,\alpha^{k-1},0,\ldots,0),\ldots,(0,\ldots 0.1)$.
- The basis reduction algorithm can be used to find a small vector in L_k, i.e. a vector of rational integers h$g_0,\ldots,g_k$i such that $|g_k\alpha^k+g_{k-1}\alpha^{k-1}+\ldots+g_0| = \varepsilon_k$ is small.
- If ε_k is sufficiently small and α is sufficiently accurate, then we have an irreducible divisor of u, namely
- $g(x)= g_k x^k+g_{k-1}x^{k-1}+\ldots+g_0$

Difficulties:

- Not all polys have real zeroes.
- Using high-precision floats can be painful.
- The actual cost of L^3 may be, and apparently in practice IS, higher than the actual cost of the exponential-worst-case (Berlekamp + Hensel) algorithm. (Especially if we have tried several primes to cut down on the number of spurious factors).

In reality what is proposed is different, but also lattice based.

- We really have factors mod p, p^2, ...p^k. Too many of them. It might take exponential time to fit them together, and we can do better by observing that the set of polynomials in Z[x] of degree less than or equal to some r that divide u(x) mod p^m form a lattice. The Lovasz reduction algorithm allows us to find a short vector in this lattice which will correspond to a factor of u. (details, e.g. in Zippel..)