Lecture 10: Univariate Polynomial Factoring over the Integers

Rafael Oliveira

University of Waterloo
Cheriton School of Computer Science

rafael.oliveira.teaching@gmail.com

February 10, 2021

Overview

- Review from last lecture: Cantor-Zassenhaus
- Today's algorithm: Berlekamp's algorithm (1967)
- Properties of Irreducible Polynomials
- Conclusion
- Acknowledgements

Problem Definition

- We know that $\mathbb{Z}[x]$ is a UFD, by Gauss' lemma. Thus each polynomial $f(x) \in \mathbb{Z}[x]$ can be factored as

$$f(x) = c_1 \cdots c_k \cdot f_1(x) \cdots f_t(x)$$

Problem Definition

- We know that $\mathbb{Z}[x]$ is a UFD, by Gauss' lemma. Thus each polynomial $f(x) \in \mathbb{Z}[x]$ can be factored as

$$f(x) = c_1 \cdots c_k \cdot f_1(x) \cdots f_t(x)$$

- Since integer factoring is hard, we will relax our problem as follows: output

$$f(x) = c \cdot f_1(x) \cdots f_t(x)$$

- **Input:** polynomial $f(x) \in \mathbb{Z}[x]$
- **Output:** Either f is irreducible or a non-trivial factorization of f

Problem Definition

- We know that $\mathbb{Z}[x]$ is a UFD, by Gauss' lemma. Thus each polynomial $f(x) \in \mathbb{Z}[x]$ can be factored as
$$f(x) = c_1 \cdots c_k \cdot f_1(x) \cdots f_t(x)$$
- Since integer factoring is hard, we will relax our problem as follows: output
$$f(x) = c \cdot f_1(x) \cdots f_t(x)$$
- **Input:** polynomial $f(x) \in \mathbb{Z}[x]$
- **Output:** Either f is irreducible or a non-trivial factorization of f
- Today we will see a *deterministic* polynomial time algorithm for integer factoring
- Factoring polynomials over the rationals can be reduced to integer factoring, by clearing denominators

Problem Definition

- We know that $\mathbb{Z}[x]$ is a UFD, by Gauss' lemma. Thus each polynomial $f(x) \in \mathbb{Z}[x]$ can be factored as

$$f(x) = c_1 \cdots c_k \cdot f_1(x) \cdots f_t(x)$$

- Since integer factoring is hard, we will relax our problem as follows: output

$$f(x) = c \cdot f_1(x) \cdots f_t(x)$$

- **Input:** polynomial $f(x) \in \mathbb{Z}[x]$
- **Output:** Either f is irreducible or a non-trivial factorization of f
- Today we will see a *deterministic* polynomial time algorithm for integer factoring
- Factoring polynomials over the rationals can be reduced to integer factoring, by clearing denominators
- One approach is to factor $f(x)$ mod p for many primes p and then see if these factorizations give us anything about the factorization of f over $\mathbb{Z}[x]$

Problem Definition

- We know that $\mathbb{Z}[x]$ is a UFD, by Gauss' lemma. Thus each polynomial $f(x) \in \mathbb{Z}[x]$ can be factored as

$$f(x) = c_1 \cdots c_k \cdot f_1(x) \cdots f_t(x)$$

- Since integer factoring is hard, we will relax our problem as follows: output

$$f(x) = c \cdot f_1(x) \cdots f_t(x)$$

- **Input:** polynomial $f(x) \in \mathbb{Z}[x]$
- **Output:** Either f is irreducible or a non-trivial factorization of f
- Today we will see a *deterministic* polynomial time algorithm for integer factoring
- Factoring polynomials over the rationals can be reduced to integer factoring, by clearing denominators
- One approach is to factor $f(x)$ mod p for many primes p and then see if these factorizations give us anything about the factorization of f over $\mathbb{Z}[x]$
- Counterexample: $f(x) = x^4 + 1$ is *irreducible* over $\mathbb{Z}[x]$ but factors over $\mathbb{Z}_p[x]$ for *any* prime p

Counterexample to First Approach

- $f(x) = x^4 + 1$
- Eisenstein's criterion over $f(x+1)$ gives us irreducibility
  1. prime p divides all coefficients but leading coefficient
  2. p^2 does not divide constant term

Counterexample to First Approach

- $f(x) = x^4 + 1$
- Eisenstein's criterion over $f(x+1)$ gives us irreducibility
  1. prime p divides all coefficients but leading coefficient
  2. p^2 does not divide constant term
- If $p = 2$, we have $x^4 + 1 = (x+1)^4$

Counterexample to First Approach

- $f(x) = x^4 + 1$
- Eisenstein's criterion over $f(x+1)$ gives us irreducibility
  1. prime p divides all coefficients but leading coefficient
  2. p^2 does not divide constant term
- If $p = 2$, we have $x^4 + 1 = (x+1)^4$
- If p odd, then $8 \mid p^2 - 1$

Counterexample to First Approach

- $f(x) = x^4 + 1$
- Eisenstein's criterion over $f(x+1)$ gives us irreducibility
  1. prime p divides all coefficients but leading coefficient
  2. p^2 does not divide constant term
- If $p = 2$, we have $x^4 + 1 = (x+1)^4$
- If p odd, then $8 \mid p^2 - 1$
- Take primitive root of unity u over $\mathbb{F}_{p^2}$. Let $g = u^{(p^2-1)/8}$. Must have $g^4 + 1 = 0$ over $\mathbb{F}_{p^2}$

Counterexample to First Approach

- $f(x) = x^4 + 1$
- Eisenstein's criterion over $f(x+1)$ gives us irreducibility
  1. prime p divides all coefficients but leading coefficient
  2. p^2 does not divide constant term
- If $p = 2$, we have $x^4 + 1 = (x+1)^4$
- If p odd, then $8 \mid p^2 - 1$
- Take primitive root of unity u over $\mathbb{F}_{p^2}$. Let $g = u^{(p^2-1)/8}$. Must have $g^4 + 1 = 0$ over $\mathbb{F}_{p^2}$
- Thus, we have

$$x^4 + 1 = (x-g)(x-g^3)(x-g^5)(x-g^7)$$

Counterexample to First Approach

- $f(x) = x^4 + 1$
- Eisenstein's criterion over $f(x+1)$ gives us irreducibility
  1. prime p divides all coefficients but leading coefficient
  2. p^2 does not divide constant term
- If $p = 2$, we have $x^4 + 1 = (x+1)^4$
- If p odd, then $8 \mid p^2 - 1$
- Take primitive root of unity u over $\mathbb{F}_{p^2}$. Let $g = u^{(p^2-1)/8}$. Must have $g^4 + 1 = 0$ over $\mathbb{F}_{p^2}$
- Thus, we have

$$x^4 + 1 = (x-g)(x-g^3)(x-g^5)(x-g^7)$$

- $g \in \mathbb{F}_{p^2}$ implies that the *minimal polynomial* of g is of degree ≤ 2 and it must divide $x^4 + 1$

Another proof of counterexample

- If $p = 2$, we have $x^4 + 1 = (x + 1)^4$
- If p odd, then we have two cases:

Another proof of counterexample

- If $p = 2$, we have $x^4 + 1 = (x + 1)^4$
- If p odd, then we have two cases:
- -1 is a square over $\mathbb{Z}_p$, say $-1 = a^2$

Another proof of counterexample

- If $p = 2$, we have $x^4 + 1 = (x + 1)^4$
- If p odd, then we have two cases:
- -1 is a square over $\mathbb{Z}_p$, say $-1 = a^2$
- In this case $x^4 + 1 = (x^2 - a)(x^2 + a)$

Another proof of counterexample

- If $p = 2$, we have $x^4 + 1 = (x + 1)^4$
- If p odd, then we have two cases:
- -1 is a square over $\mathbb{Z}_p$, say $-1 = a^2$
- In this case $x^4 + 1 = (x^2 - a)(x^2 + a)$
- 2 is a square over $\mathbb{Z}_p$, say $2 = b^2$

Another proof of counterexample

- If $p = 2$, we have $x^4 + 1 = (x + 1)^4$
- If p odd, then we have two cases:
- -1 is a square over $\mathbb{Z}_p$, say $-1 = a^2$
- In this case $x^4 + 1 = (x^2 - a)(x^2 + a)$
- 2 is a square over $\mathbb{Z}_p$, say $2 = b^2$
- Thus, we have

$$x^4 + 1 + 2x^2 = (x^2 + 1)^2 \Rightarrow x^4 + 1 = (x^2 + 1)^2 - 2x^2$$
$$= (x^2 - bx + 1)(x^2 + bx + 1)$$

Another proof of counterexample

- If $p = 2$, we have $x^4 + 1 = (x + 1)^4$
- If p odd, then we have two cases:
- -1 is a square over $\mathbb{Z}_p$, say $-1 = a^2$
- In this case $x^4 + 1 = (x^2 - a)(x^2 + a)$
- 2 is a square over $\mathbb{Z}_p$, say $2 = b^2$
- Thus, we have

$$x^4 + 1 + 2x^2 = (x^2 + 1)^2 \Rightarrow x^4 + 1 = (x^2 + 1)^2 - 2x^2$$
$$= (x^2 - bx + 1)(x^2 + bx + 1)$$

- If -1 nor 2 are a square over $\mathbb{Z}_p$, then $-2 = c^2$

Another proof of counterexample

- If $p = 2$, we have $x^4 + 1 = (x + 1)^4$
- If p odd, then we have two cases:
- -1 is a square over $\mathbb{Z}_p$, say $-1 = a^2$
- In this case $x^4 + 1 = (x^2 - a)(x^2 + a)$
- 2 is a square over $\mathbb{Z}_p$, say $2 = b^2$
- Thus, we have

$$x^4 + 1 + 2x^2 = (x^2 + 1)^2 \Rightarrow x^4 + 1 = (x^2 + 1)^2 - 2x^2$$
$$= (x^2 - bx + 1)(x^2 + bx + 1)$$

- If -1 nor 2 are a square over $\mathbb{Z}_p$, then $-2 = c^2$
- Thus we have

$$x^4 + 1 - 2x^2 = (x^2 - 1)^2 \Rightarrow x^4 + 1 = (x^2 + 1)^2 + 2x^2$$
$$= (x^2 - cx + 1)(x^2 + cx + 1)$$

Factoring Algorithm

- Another approach is to pick a *good prime* p and then factor $f(x)$ modulo p^k for large enough k. We can hope that the large modulus will give us a factorization over $\mathbb{Z}$, if it exists.

Factoring Algorithm

- Another approach is to pick a *good prime* p and then factor $f(x)$ modulo p^k for large enough k. We can hope that the large modulus will give us a factorization over $\mathbb{Z}$, if it exists.
- **Input:** polynomial $f(x) = a_0 + a_1 x + \cdots + a_d x^d \in \mathbb{Z}[x]$, where a_i has ℓ bits
- **Output:** Either f is irreducible or a non-trivial factorization of f

Factoring Algorithm

- Another approach is to pick a *good prime* p and then factor $f(x)$ modulo p^k for large enough k. We can hope that the large modulus will give us a factorization over $\mathbb{Z}$, if it exists.
- **Input:** polynomial $f(x) = a_0 + a_1 x + \cdots + a_d x^d \in \mathbb{Z}[x]$, where a_i has ℓ bits
- **Output:** Either f is irreducible or a non-trivial factorization of f
- Reduce to the case f is square-free, by replacing f with $\dfrac{f}{\gcd(f, f')}$

Factoring Algorithm

- Another approach is to pick a *good prime* p and then factor $f(x)$ modulo p^k for large enough k. We can hope that the large modulus will give us a factorization over $\mathbb{Z}$, if it exists.
- **Input:** polynomial $f(x) = a_0 + a_1 x + \cdots + a_d x^d \in \mathbb{Z}[x]$, where a_i has ℓ bits
- **Output:** Either f is irreducible or a non-trivial factorization of f
- Reduce to the case f is square-free, by replacing f with $\dfrac{f}{\gcd(f, f')}$
- Let p be a prime which does not divide a_d

Factoring Algorithm

- Another approach is to pick a *good prime* p and then factor $f(x)$ modulo p^k for large enough k. We can hope that the large modulus will give us a factorization over $\mathbb{Z}$, if it exists.
- **Input:** polynomial $f(x) = a_0 + a_1 x + \cdots + a_d x^d \in \mathbb{Z}[x]$, where a_i has ℓ bits
- **Output:** Either f is irreducible or a non-trivial factorization of f
- Reduce to the case f is square-free, by replacing f with $\dfrac{f}{\gcd(f, f')}$
- Let p be a prime which does not divide a_d
- Factor $f(x) = g_0(x) \cdot h_0(x) \mod p$ where g_0 is *irreducible, monic* and *relatively prime* to h_0

Factoring Algorithm

- Another approach is to pick a *good prime* p and then factor $f(x)$ modulo p^k for large enough k. We can hope that the large modulus will give us a factorization over $\mathbb{Z}$, if it exists.
- **Input:** polynomial $f(x) = a_0 + a_1 x + \cdots + a_d x^d \in \mathbb{Z}[x]$, where a_i has ℓ bits
- **Output:** Either f is irreducible or a non-trivial factorization of f
- Reduce to the case f is square-free, by replacing f with $\dfrac{f}{\gcd(f, f')}$
- Let p be a prime which does not divide a_d
- Factor $f(x) = g_0(x) \cdot h_0(x) \mod p$ where g_0 is *irreducible, monic* and *relatively prime* to h_0
- How can we get a *consistent* factorization modulo p^2

Hensel's lifting lemma.

Hensel Lifting

- If R is a UFD and $I \subseteq R$ be an ideal. For any $f \in R$ and for any factorization $f = gh \mod I$ such that there are $a, b \in R$ for which

$$ag + bh = 1 \mod I$$

then we can find $G, H, A, B \in R$ such that

$$f = GH \mod I^2 \quad \text{and} \quad AG + BH = 1 \mod I^2$$

and

$$G = g \mod I \quad \text{and} \quad H = h \mod I$$

Hensel Lifting

- If R is a UFD and $I \subseteq R$ be an ideal. For any $f \in R$ and for any factorization $f = gh \mod I$ such that there are $a, b \in R$ for which

$$ag + bh = 1 \mod I$$

then we can find $G, H, A, B \in R$ such that

$$f = GH \mod I^2 \quad \text{and} \quad AG + BH = 1 \mod I^2$$

and

$$G = g \mod I \quad \text{and} \quad H = h \mod I$$

- Moreover, solution above is unique, in the following sense: if G_1, H_1 also have the properties above, then there exists $u \in I$ such that

$$G_1 = G(1 + u) \mod I^2 \quad \text{and} \quad H_1 = H(1 - u) \mod I^2$$

Factoring Algorithm

- **Input:** $f(x) = a_0 + a_1 x + \cdots + a_d x^d \in \mathbb{Z}[x]$, where a_i has ℓ bits
- **Output:** Either f is irreducible or a non-trivial factorization of f

Factoring Algorithm

- **Input:** $f(x) = a_0 + a_1 x + \cdots + a_d x^d \in \mathbb{Z}[x]$, where a_i has ℓ bits
- **Output:** Either f is irreducible or a non-trivial factorization of f
- Reduce to the case f is square-free, by replacing f with $\frac{f}{\gcd(f, f')}$

Factoring Algorithm

- **Input:** $f(x) = a_0 + a_1 x + \cdots + a_d x^d \in \mathbb{Z}[x]$, where a_i has ℓ bits
- **Output:** Either f is irreducible or a non-trivial factorization of f
- Reduce to the case f is square-free, by replacing f with $\dfrac{f}{\gcd(f, f')}$
- Let p be a prime which does not divide a_d

Factoring Algorithm

- **Input:** $f(x) = a_0 + a_1 x + \cdots + a_d x^d \in \mathbb{Z}[x]$, where a_i has ℓ bits
- **Output:** Either f is irreducible or a non-trivial factorization of f
- Reduce to the case f is square-free, by replacing f with $\dfrac{f}{\gcd(f, f')}$
- Let p be a prime which does not divide a_d
- Factor $f(x) = g_0(x) \cdot h_0(x) \mod p$ where g_0 is *irreducible, monic* and *relatively prime* to h_0

Factoring Algorithm

- **Input:** $f(x) = a_0 + a_1 x + \cdots + a_d x^d \in \mathbb{Z}[x]$, where a_i has ℓ bits
- **Output:** Either f is irreducible or a non-trivial factorization of f
- Reduce to the case f is square-free, by replacing f with $\frac{f}{\gcd(f, f')}$
- Let p be a prime which does not divide a_d
- Factor $f(x) = g_0(x) \cdot h_0(x) \mod p$ where g_0 is *irreducible, monic* and *relatively prime* to h_0
- Iteratively use Hensel Lifting to get factorization $f = g_k h_k \mod p^k$

Factoring Algorithm

- **Input:** $f(x) = a_0 + a_1 x + \cdots + a_d x^d \in \mathbb{Z}[x]$, where a_i has ℓ bits
- **Output:** Either f is irreducible or a non-trivial factorization of f
- Reduce to the case f is square-free, by replacing f with $\dfrac{f}{\gcd(f, f')}$
- Let p be a prime which does not divide a_d
- Factor $f(x) = g_0(x) \cdot h_0(x) \mod p$ where g_0 is *irreducible, monic* and *relatively prime* to h_0
- Iteratively use Hensel Lifting to get factorization $f = g_k h_k \mod p^k$
- Find $p(x)$ and $q_k(x)$ such that $p(x)$ has "small coefficients" and:

$$p(x) = g_k(x) q_k(x) \mod p^k$$
$$\deg(p) \leq \deg(f)$$

Factoring Algorithm

- **Input:** $f(x) = a_0 + a_1 x + \cdots + a_d x^d \in \mathbb{Z}[x]$, where a_i has ℓ bits
- **Output:** Either f is irreducible or a non-trivial factorization of f
- Reduce to the case f is square-free, by replacing f with $\dfrac{f}{\gcd(f, f')}$
- Let p be a prime which does not divide a_d
- Factor $f(x) = g_0(x) \cdot h_0(x) \mod p$ where g_0 is *irreducible, monic* and *relatively prime* to h_0
- Iteratively use Hensel Lifting to get factorization $f = g_k h_k \mod p^k$
- Find $p(x)$ and $q_k(x)$ such that $p(x)$ has "small coefficients" and:

$$p(x) = g_k(x) q_k(x) \mod p^k$$
$$\deg(p) \le \deg(f)$$

- if $\gcd(f, p)$ non trivial, output $\gcd(f, p)$ and $\dfrac{f}{\gcd f, p}$. Otherwise, output irreducible.

- Review from last lecture: Cantor-Zassenhaus
- Today's algorithm: Berlekamp's algorithm (1967)
- Properties of Irreducible Polynomials
- Conclusion
- Acknowledgements

Bounds on coeficient size of factors of f

- If $f(x) = a_0 + a_1 x + \cdots + a_d x^d \in \mathbb{Z}[x]$, where a_i has ℓ bits and $g(x) \mid f(x)$ then the coefficients of g have at most poly(d, ℓ) bits

Bounds on coeficient size of factors of f

- If $f(x) = a_0 + a_1 x + \cdots + a_d x^d \in \mathbb{Z}[x]$, where a_i has ℓ bits and $g(x) \mid f(x)$ then the coefficients of g have at most poly(d, ℓ) bits
- To see this, note that any (complex) root of $f(x)$ do not have large absolute value: if $\alpha \in \mathbb{C}$ such that $f(\alpha) = 0$ then $|\alpha| \leq d \cdot 2^{\ell}$

Bounds on coeficient size of factors of f

- If $f(x) = a_0 + a_1 x + \cdots + a_d x^d \in \mathbb{Z}[x]$, where a_i has ℓ bits and $g(x) \mid f(x)$ then the coefficients of g have at most $\text{poly}(d, \ell)$ bits
- To see this, note that any (complex) root of $f(x)$ do not have large absolute value: if $\alpha \in \mathbb{C}$ such that $f(\alpha) = 0$ then $|\alpha| \leq d \cdot 2^\ell$
- Suppose that $|\alpha| > d \cdot 2^\ell$. In particular, $|\alpha| > |a_i|$ for any $0 \leq i \leq d$

Bounds on coeficient size of factors of f

- If $f(x) = a_0 + a_1 x + \cdots + a_d x^d \in \mathbb{Z}[x]$, where a_i has ℓ bits and $g(x) \mid f(x)$ then the coefficients of g have at most poly(d, ℓ) bits
- To see this, note that any (complex) root of $f(x)$ do not have large absolute value: if $\alpha \in \mathbb{C}$ such that $f(\alpha) = 0$ then $|\alpha| \leq d \cdot 2^\ell$
- Suppose that $|\alpha| > d \cdot 2^\ell$. In particular, $|\alpha| > |a_i|$ for any $0 \leq i \leq d$
- Thus

$$\begin{aligned}
|f(\alpha)| &= \left| a_d \alpha^d + \sum_{i=0}^{d-1} a_i \alpha^i \right| \\
&\geq |\alpha|^d - d \cdot 2^\ell |\alpha|^{d-1} \\
&= |\alpha|^{d-1} \cdot (|\alpha| - d \cdot 2^\ell) > 0
\end{aligned}$$

Bounds on coeficient size of factors of f

- If $f(x) = a_0 + a_1 x + \cdots + a_d x^d \in \mathbb{Z}[x]$, where a_i has ℓ bits and $g(x) \mid f(x)$ then the coefficients of g have at most poly(d, ℓ) bits
- To see this, note that any (complex) root of $f(x)$ do not have large absolute value: if $\alpha \in \mathbb{C}$ such that $f(\alpha) = 0$ then $|\alpha| \leq d \cdot 2^\ell$
- Suppose that $|\alpha| > d \cdot 2^\ell$. In particular, $|\alpha| > |a_i|$ for any $0 \leq i \leq d$
- Thus

$$
\begin{aligned}
|f(\alpha)| &= \left| a_d \alpha^d + \sum_{i=0}^{d-1} a_i \alpha^i \right| \\
&\geq |\alpha|^d - d \cdot 2^\ell |\alpha|^{d-1} \\
&= |\alpha|^{d-1} \cdot (|\alpha| - d \cdot 2^\ell) > 0
\end{aligned}
$$

- Since $g(x) = b \cdot \prod_{i \in S}(x - \alpha_i)$ where $b \mid a_d$ and S subset of roots of f, we have that coefficients of g are upper bounded in absolute value by $2^{\ell+d} \cdot (d \cdot 2^\ell)^d$

Bound on coefficients of g

Factoring Algorithm

- **Input:** $f(x) = a_0 + a_1 x + \cdots + a_d x^d \in \mathbb{Z}[x]$, where a_i has ℓ bits
- **Output:** Either f is irreducible or a non-trivial factorization of f
- Reduce to the case f is square-free, by replacing f with $\dfrac{f}{\gcd(f, f')}$
- Let p be a prime which does not divide a_d
- Factor $f(x) = g_0(x) \cdot h_0(x) \mod p$ where g_0 is *irreducible, monic* and *relatively prime* to h_0
- Iteratively use Hensel Lifting to get factorization $f = g_k h_k \mod p^k$
- Find $p(x)$ and $q_k(x)$ such that $p(x)$ has "small coefficients" and:

$$p(x) = g_k(x) q_k(x) \mod p^k$$
$$\deg(p) \leq \deg(f)$$

- if $\gcd(f, p)$ non trivial, output $\gcd(f, p)$ and $\dfrac{f}{\gcd f, p}$. Otherwise, output irreducible.

How do we find g_k, h_k, p and q_k

- If f factors, it must still factor modulo the prime we chose, as p does not divide a_d

How do we find g_k, h_k, p and q_k

- If f factors, it must still factor modulo the prime we chose, as p does not divide a_d
- We know that g_k, h_k exist by Hensel Lifting

How do we find g_k, h_k, p and q_k

- If f factors, it must still factor modulo the prime we chose, as p does not divide a_d
- We know that g_k, h_k exist by Hensel Lifting
- How can we find p, q_k?

$$p(x) = g_k(x)q_k(x) \mod p^k$$
$$\deg(p) \leq \deg(f)$$

By solving the linear system above.

How do we find g_k, h_k, p and q_k

- If f factors, it must still factor modulo the prime we chose, as p does not divide a_d
- We know that g_k, h_k exist by Hensel Lifting
- How can we find p, q_k?

$$p(x) = g_k(x)q_k(x) \mod p^k$$
$$\deg(p) \leq \deg(f)$$

  By solving the linear system above.
- Why would such a p with small coefficients exist? In particular, the factor of f which has g_0 as a factor has "small" coefficients, by our previous lemma.

How do we find g_k, h_k, p and q_k

- If f factors, it must still factor modulo the prime we chose, as p does not divide a_d
- We know that g_k, h_k exist by Hensel Lifting
- How can we find p, q_k?

$$p(x) = g_k(x)q_k(x) \mod p^k$$
$$\deg(p) \leq \deg(f)$$

  By solving the linear system above.
- Why would such a p with small coefficients exist? In particular, the factor of f which has g_0 as a factor has "small" coefficients, by our previous lemma.
- How do we find a polynomial p with "small" coefficients though? (will see this next section and lecture)

How do we find g_k, h_k, p and q_k

- If f factors, it must still factor modulo the prime we chose, as p does not divide a_d
- We know that g_k, h_k exist by Hensel Lifting
- How can we find p, q_k?

$$p(x) = g_k(x)q_k(x) \mod p^k$$
$$\deg(p) \leq \deg(f)$$

  By solving the linear system above.
- Why would such a p with small coefficients exist? In particular, the factor of f which has g_0 as a factor has "small" coefficients, by our previous lemma.
- How do we find a polynomial p with "small" coefficients though? (will see this next section and lecture)
- Suppose we can find p with small coefficient, do we have non-trivial GCD?

Non-trivial GCD

- If f, p are integer polynomials satisfying the conditions of the algorithm, with coefficients having at most B bits, and $p^k > (2d)! \cdot 2^{2dB}$ then $\gcd(f, p)$ is non-trivial

Non-trivial GCD

- If f, p are integer polynomials satisfying the conditions of the algorithm, with coefficients having at most B bits, and $p^k > (2d)! \cdot 2^{2dB}$ then $\gcd(f, p)$ is non-trivial
- Since $\deg(p) < \deg(f)$, we have $\gcd(p, f) \neq f$. Suppose $\gcd(p, f) = 1$ over $\mathbb{Q}[x]$, for sake of contradiction

Non-trivial GCD

- If f, p are integer polynomials satisfying the conditions of the algorithm, with coefficients having at most B bits, and $p^k > (2d)! \cdot 2^{2dB}$ then $\gcd(f, p)$ is non-trivial
- Since $\deg(p) < \deg(f)$, we have $\gcd(p, f) \neq f$. Suppose $\gcd(p, f) = 1$ over $\mathbb{Q}[x]$, for sake of contradiction
- Then, there are polynomials $s, t \in \mathbb{Z}[x]$ and $N \in \mathbb{Z} \setminus \{0\}$ such that

$$sf + tp = N \quad \text{and} \quad N \leq \text{Res}_x(f, p)$$

Non-trivial GCD

- If f, p are integer polynomials satisfying the conditions of the algorithm, with coefficients having at most B bits, and $p^k > (2d)! \cdot 2^{2dB}$ then $\gcd(f, p)$ is non-trivial
- Since $\deg(p) < \deg(f)$, we have $\gcd(p, f) \neq f$. Suppose $\gcd(p, f) = 1$ over $\mathbb{Q}[x]$, for sake of contradiction
- Then, there are polynomials $s, t \in \mathbb{Z}[x]$ and $N \in \mathbb{Z} \setminus \{0\}$ such that

$$sf + tp = N \quad \text{and} \quad N \leq \text{Res}_x(f, p)$$

- From Lecture 7's bound, we have $N \leq (2d)! \cdot 2^{2dB}$. Thus, $p^k > N$, which implies $N \neq 0 \mod p^k$

Non-trivial GCD

- If f, p are integer polynomials satisfying the conditions of the algorithm, with coefficients having at most B bits, and $p^k > (2d)! \cdot 2^{2dB}$ then $\gcd(f, p)$ is non-trivial
- Since $\deg(p) < \deg(f)$, we have $\gcd(p, f) \neq f$. Suppose $\gcd(p, f) = 1$ over $\mathbb{Q}[x]$, for sake of contradiction
- Then, there are polynomials $s, t \in \mathbb{Z}[x]$ and $N \in \mathbb{Z} \setminus \{0\}$ such that

$$sf + tp = N \quad \text{and} \quad N \leq \mathrm{Res}_x(f, p)$$

- From Lecture 7's bound, we have $N \leq (2d)! \cdot 2^{2dB}$. Thus, $p^k > N$, which implies $N \not\equiv 0 \mod p^k$
- Thus, we would have

$$\begin{aligned} N &= sf + tp \mod p^k \\ &= s(g_k h_k) + t(g_k \cdot q_k) \mod p^k \\ &= g_k \cdot (sh_k + tq_k) \mod p^k \end{aligned}$$

which is a contradiction, since $\deg(g_k) \geq 1$

- Review from last lecture: Cantor-Zassenhaus
- Today's algorithm: Berlekamp's algorithm (1967)
- **Properties of Irreducible Polynomials**
- Conclusion
- Acknowledgements

Factoring Algorithm

- Let p be a prime which does not divide a_d
- Factor $f(x) = g_0(x) \cdot h_0(x) \mod p$ where g_0 is *irreducible, monic* and *relatively prime* to h_0
- Iteratively use Hensel Lifting to get factorization $f = g_k h_k \mod p^k$
- Find $p(x)$ and $q_k(x)$ such that $p(x)$ has "small coefficients" and:

$$p(x) = g_k(x) q_k(x) \mod p^k$$
$$\deg(p) \leq \deg(f)$$

- if $\gcd(f, p)$ non trivial, output $\gcd(f, p)$ and $\dfrac{f}{\gcd f, p}$. Otherwise, output irreducible.

Factoring Algorithm

- Let p be a prime which does not divide a_d
- Factor $f(x) = g_0(x) \cdot h_0(x) \mod p$ where g_0 is *irreducible, monic* and *relatively prime* to h_0
- Iteratively use Hensel Lifting to get factorization $f = g_k h_k \mod p^k$
- Find $p(x)$ and $q_k(x)$ such that $p(x)$ has "small coefficients" and:

$$p(x) = g_k(x) q_k(x) \mod p^k$$
$$\deg(p) \leq \deg(f)$$

- if $\gcd(f,p)$ non trivial, output $\gcd(f,p)$ and $\dfrac{f}{\gcd f, p}$. Otherwise, output irreducible.
- We know such a factor p must exist, if f factors
- All we need (by previous lemma) is to find *some solution* with *small enough* height.

Factoring Algorithm

- Let p be a prime which does not divide a_d
- Factor $f(x) = g_0(x) \cdot h_0(x) \mod p$ where g_0 is *irreducible, monic* and *relatively prime* to h_0
- Iteratively use Hensel Lifting to get factorization $f = g_k h_k \mod p^k$
- Find $p(x)$ and $q_k(x)$ such that $p(x)$ has "small coefficients" and:

$$p(x) = g_k(x) q_k(x) \mod p^k$$
$$\deg(p) \leq \deg(f)$$

- if $\gcd(f, p)$ non trivial, output $\gcd(f, p)$ and $\frac{f}{\gcd f, p}$. Otherwise, output irreducible.
- We know such a factor p must exist, if f factors
- All we need (by previous lemma) is to find *some solution* with *small enough* height.
- This problem is exactly the problem of finding a *small vector in a lattice*

Small Vectors in a Lattice

- **Input:** linearly independent vectors $b_1, \ldots, b_m \in \mathbb{Z}^n$, bound $M \in \mathbb{N}$

$$\mathcal{L} = \{\alpha_1 b_1 + \cdots \alpha_n b_m \mid \alpha_i \in \mathbb{Z}\}$$

- **Output:** A vector $v \in \mathcal{L}$ such that $\|v\| \leq M$

Small Vectors in a Lattice

- **Input:** linearly independent vectors $b_1, \ldots, b_m \in \mathbb{Z}^n$, bound $M \in \mathbb{N}$

$$\mathcal{L} = \{\alpha_1 b_1 + \cdots \alpha_n b_m \mid \alpha_i \in \mathbb{Z}\}$$

- **Output:** A vector $v \in \mathcal{L}$ such that $\|v\| \leq M$
- Note that to finish the factoring problem, compute a basis for the set of integral solutions $(p(x), q(x))$ to

$$\begin{aligned} p(x) &= g_k(x) q_k(x) \mod p^k \\ \deg(p) &\leq \deg(f) \end{aligned}$$

Small Vectors in a Lattice

- **Input:** linearly independent vectors $b_1, \ldots, b_m \in \mathbb{Z}^n$, bound $M \in \mathbb{N}$

$$\mathcal{L} = \{\alpha_1 b_1 + \cdots \alpha_n b_m \mid \alpha_i \in \mathbb{Z}\}$$

- **Output:** A vector $v \in \mathcal{L}$ such that $\|v\| \leq M$
- Note that to finish the factoring problem, compute a basis for the set of integral solutions $(p(x), q(x))$ to

$$\begin{aligned} p(x) &= g_k(x) q_k(x) \mod p^k \\ \deg(p) &\leq \deg(f) \end{aligned}$$

- Make solutions integral by adding $\beta_i p^k x^i$ for $0 \leq i \leq \deg(p)$

Small Vectors in a Lattice

- **Input:** linearly independent vectors $b_1, \ldots, b_m \in \mathbb{Z}^n$, bound $M \in \mathbb{N}$

$$\mathcal{L} = \{\alpha_1 b_1 + \cdots \alpha_n b_m \mid \alpha_i \in \mathbb{Z}\}$$

- **Output:** A vector $v \in \mathcal{L}$ such that $\|v\| \leq M$
- Note that to finish the factoring problem, compute a basis for the set of integral solutions $(p(x), q(x))$ to

$$\begin{aligned} p(x) &= g_k(x)q_k(x) \mod p^k \\ \deg(p) &\leq \deg(f) \end{aligned}$$

- Make solutions integral by adding $\beta_i p^k x^i$ for $0 \leq i \leq \deg(p)$
- these integral solutions form a lattice, and we can find a basis for this lattice

Small Vectors in a Lattice

- **Input:** linearly independent vectors $b_1, \ldots, b_m \in \mathbb{Z}^n$, bound $M \in \mathbb{N}$

$$\mathcal{L} = \{\alpha_1 b_1 + \cdots \alpha_n b_m \mid \alpha_i \in \mathbb{Z}\}$$

- **Output:** A vector $v \in \mathcal{L}$ such that $\|v\| \leq M$
- Note that to finish the factoring problem, compute a basis for the set of integral solutions $(p(x), q(x))$ to

$$p(x) = g_k(x) q_k(x) \mod p^k$$
$$\deg(p) \leq \deg(f)$$

- Make solutions integral by adding $\beta_i p^k x^i$ for $0 \leq i \leq \deg(p)$
- these integral solutions form a lattice, and we can find a basis for this lattice
- the small vectors in a lattice problem above helps us find the polynomial p that we want.

Conclusion

In today's lecture, we learned

- Factoring algorithm for integer polynomials
- CRT doesn't work
- Need to use Hensel lifting instead (generalization of Newton's method)
- Reduced factoring problem to the problem of finding a small vector in a lattice *next lecture*

Acknowledgement

Based entirely on

- Lecture 10 from Madhu's notes
  http://people.csail.mit.edu/madhu/FT98/