Lecture 7: Resultants & Modular GCD algorithm

Rafael Oliveira

University of Waterloo
Cheriton School of Computer Science

rafael.oliveira.teaching@gmail.com

January 31, 2021

Overview

- Resultants & Discriminants
- Modular GCD algorithm in $\mathbb{Z}[x]$
- Conclusion
- Acknowledgements

- Resultants & Discriminants
- Modular GCD algorithm in $\mathbb{Z}[x]$
- Conclusion
- Acknowledgements

Unique Factorization Domains

- An integral domain R is a *unique factorization domain* (UFD) if
  1. every element in R is expressed as a product of finitely many irreducible elements
  2. Every irreducible element $p \in R$ yields a prime ideal (p)

Unique Factorization Domains

- An integral domain R is a *unique factorization domain* (UFD) if
  1. every element in R is expressed as a product of finitely many irreducible elements
  2. Every irreducible element $p \in R$ yields a prime ideal (p)
- A very special kind of UFD, which we have seen a lot, is a *principal ideal domain* (PID): R is a PID if every ideal of R is principal (generated by *one element*)

Unique Factorization Domains

- An integral domain R is a *unique factorization domain* (UFD) if
  1. every element in R is expressed as a product of finitely many irreducible elements
  2. Every irreducible element $p \in R$ yields a prime ideal (p)
- A very special kind of UFD, which we have seen a lot, is a *principal ideal domain* (PID): R is a PID if every ideal of R is principal (generated by *one element*)
- Examples of PIDs and UFDs
  1. $\mathbb{Z}$ is a PID (and hence UFD)

Unique Factorization Domains

- An integral domain R is a *unique factorization domain* (UFD) if
  1. every element in R is expressed as a product of finitely many irreducible elements
  2. Every irreducible element $p \in R$ yields a prime ideal (p)
- A very special kind of UFD, which we have seen a lot, is a *principal ideal domain* (PID): R is a PID if every ideal of R is principal (generated by *one element*)
- Examples of PIDs and UFDs
  1. $\mathbb{Z}$ is a PID (and hence UFD)
  2. $\mathbb{Q}[x]$ is a PID (and hence UFD)

$\mathbb{F}[x]$ PID $\quad \mathbb{F}$ field

Unique Factorization Domains

- An integral domain R is a *unique factorization domain* (UFD) if
  1. every element in R is expressed as a product of finitely many irreducible elements
  2. Every irreducible element $p \in R$ yields a prime ideal (p)
- A very special kind of UFD, which we have seen a lot, is a *principal ideal domain* (PID): R is a PID if every ideal of R is principal (generated by *one element*)
- Examples of PIDs and UFDs
  1. $\mathbb{Z}$ is a PID (and hence UFD)
  2. $\mathbb{Q}[x]$ is a PID (and hence UFD)
  3. any Euclidean domain is a PID (and hence UFD)

Unique Factorization Domains

- An integral domain R is a *unique factorization domain* (UFD) if
  1. every element in R is expressed as a product of finitely many irreducible elements
  2. Every irreducible element $p \in R$ yields a prime ideal (p)
- A very special kind of UFD, which we have seen a lot, is a *principal ideal domain* (PID): R is a PID if every ideal of R is principal (generated by *one element*)
- Examples of PIDs and UFDs
  1. $\mathbb{Z}$ is a PID (and hence UFD)
  2. $\mathbb{Q}[x]$ is a PID (and hence UFD)
  3. any Euclidean domain is a PID (and hence UFD)
  4. $\mathbb{Q}[x, y]$ is a UFD but *not* a PID

Unique Factorization Domains

- An integral domain R is a *unique factorization domain* (UFD) if
  1. every element in R is expressed as a product of finitely many irreducible elements
  2. Every irreducible element $p \in R$ yields a prime ideal (p)
- A very special kind of UFD, which we have seen a lot, is a *principal ideal domain* (PID): R is a PID if every ideal of R is principal (generated by *one element*)
- Examples of PIDs and UFDs
  1. $\mathbb{Z}$ is a PID (and hence UFD)
  2. $\mathbb{Q}[x]$ is a PID (and hence UFD)
  3. any Euclidean domain is a PID (and hence UFD)
  4. $\mathbb{Q}[x, y]$ is a UFD but *not* a PID
  5. $\mathbb{Z}[x]$ is a UFD but *not* a PID

Unique Factorization Domains

- An integral domain R is a *unique factorization domain* (UFD) if
  1. every element in R is expressed as a product of finitely many irreducible elements
  2. Every irreducible element $p \in R$ yields a prime ideal (p)
- A very special kind of UFD, which we have seen a lot, is a *principal ideal domain* (PID): R is a PID if every ideal of R is principal (generated by *one element*)
- Examples of PIDs and UFDs
  1. $\mathbb{Z}$ is a PID (and hence UFD)
  2. $\mathbb{Q}[x]$ is a PID (and hence UFD)
  3. any Euclidean domain is a PID (and hence UFD)
  4. $\mathbb{Q}[x, y]$ is a UFD but *not* a PID
  5. $\mathbb{Z}[x]$ is a UFD but *not* a PID
- Over UFDs, it makes sense to talk about *greatest common divisor* and they are very useful in symbolic computation and algebraic geometry.
  1. Factoring polynomials
  2. Irreducible components of hypersurfaces
  3. Multiplicity of roots, factors and components

Normal forms in UFDs

- Given a UFD R, let us define some "normal forms:"
  1. $\mathrm{lu} : R \to R$ "selects a unit to be special"
  2. $\mathrm{normal} : R \to R$ takes any element to its "special associate"

$$a = \underbrace{\mathrm{lu}(a)}_{\text{unit}} \cdot \mathrm{normal}(a)$$

transform any unit into the special unit (1)

Normal forms in UFDs

- Given a UFD R, let us define some "normal forms:"
  1. $\mathrm{lu} : R \to R$ "selects a unit to be special"
  2. $\mathrm{normal} : R \to R$ takes any element to its "special associate"

$$a = \mathrm{lu}(a) \cdot \mathrm{normal}(a)$$

- Examples:
  1. Over $\mathbb{Z}$, the units are $\{1, -1\}$

$$\mathrm{lu}(a) = \mathrm{sign}(a)$$

  2. Normal form over $\mathbb{Z}$ would be:

$$\mathrm{normal}(a) = |a|$$

$lu(-3) \to -1$

$lu(4) = 1$

$|-3| \to 3$

$|4| \to 4$

$-3 = lu(-3) \cdot |-3|$

Normal forms in UFDs

- Given a UFD R, let us define some "normal forms:"
  1. $\text{lu} : R \to R$ "selects a unit to be special"
  2. $\text{normal} : R \to R$ takes any element to its "special associate"

$$a = \text{lu}(a) \cdot \text{normal}(a)$$

- Examples:
  1. Over $\mathbb{Z}$, the units are $\{1, -1\}$

$$\text{lu}(a) = \text{sign}(a)$$

  2. Normal form over $\mathbb{Z}$ would be:

$$\text{normal}(a) = |a|$$

  3. Over $\mathbb{F}[x]$, the units are $\boxed{\mathbb{F} \setminus \{0\}}$

$$\text{lu}(p(x)) = LC(p)$$

  4. Normal form over $\mathbb{F}[x]$ would be the *monic polynomials*

$$\text{normal}(p(x)) = \frac{1}{LC(p)} \cdot p(x)$$

Normal forms in UFDs

- Given a UFD R, and $f(x) = f_0 + f_1 x + \cdots f_d x^d \in R[x]$, define
  1. content : $R[x] \to R$
  $$\text{content}(f) = \gcd(f_0, \ldots, f_d)$$
  gcd of coeffs.
  2. the primitive part pp : $R[x] \to R[x]$
  $$\text{pp}(f) = \frac{f}{\text{content}(f)}$$

no element of R divides pp(f)
non-unit

Normal forms in UFDs

- Given a UFD R, and $f(x) = f_0 + f_1 x + \cdots f_d x^d \in R[x]$, define
  1. content : $R[x] \to R$
  $$\mathrm{content}(f) = \gcd(f_0, \ldots, f_d)$$
  2. the primitive part pp : $R[x] \to R[x]$
  $$\mathrm{pp}(f) = \frac{f}{\mathrm{content}(f)}$$
- Example: Over $\mathbb{Z}[x]$, $f(x) = 6x^3 - 3x^2 + 9$

$\mathrm{content}(f) := \gcd(6, -3, 9) = 3$

$\mathrm{pp}(f) = 2x^3 - x^2 + 3$

Normal forms in UFDs

- Given a UFD R, and $f(x) = f_0 + f_1 x + \cdots f_d x^d \in R[x]$, define
  1. content : $R[x] \to R$
     $$\text{content}(f) = \gcd(f_0, \ldots, f_d)$$
  2. the primitive part pp : $R[x] \to R[x]$
     $$\text{pp}(f) = \frac{f}{\text{content}(f)}$$

$\mathbb{F} = \{ a/b \mid a, b \in R, b \neq 0 \}$

- Example: Over $\mathbb{Z}[x]$, $f(x) = 6x^3 - 3x^2 + 9$
- $R[x]$ is a UFD, $\mathbb{F}$ is the field of fractions of R, and

$\frac{a}{b} = \frac{a'}{b'} \Leftrightarrow ab' = a'b$

$$g(x) = (a_0/b) + (a_1/b) \cdot x + \cdots + (a_d/b) \cdot x^d \in \boxed{\mathbb{F}[x]}$$

$$\text{content}(g) = \frac{\gcd(a_0, \ldots, a_d)}{b} \quad \text{and} \quad \text{pp}(g) = \frac{g}{\text{content}(g)}$$

$\in \mathbb{F}$ $\qquad$ $\in R[x]$

Gauss' Lemma

- *Gauss' Lemma*: let R be a UFD with field of fractions $\mathbb{F}$. Then the following hold:
  1. For $f, g \in R[x]$
  $$\text{content}(fg) = \text{content}(f) \cdot \text{content}(g) \text{ and } \text{pp}(fg) = \text{pp}(f) \cdot \text{pp}(g)$$
  2. $R[x]$ is a UFD, and the unique factorization (up to units and ordering) of $f \in R[x]$ is:
  $$f(x) = \overbrace{(p_1 \cdots p_k)}^{\text{content}} \cdot \underbrace{(\text{pp}(f_1) \cdots \text{pp}(f_\ell))}_{\text{primitive part}}$$
  where
  $$\text{content}(f) = p_1 \cdots p_k \quad \text{in} \quad R$$
  and
  $$\text{pp}(f) = f_1 \cdots f_\ell \quad \text{over } \mathbb{F}[x]$$

GCD via field of fractions

- We can now compute the GCD over $R[x]$ as via it's field of fractions $\mathbb{F}[x]$, which we know is an Euclidean Domain
- Assume we can compute the GCD of two elements in R

GCD via field of fractions

- We can now compute the GCD over $R[x]$ as via it's field of fractions $\mathbb{F}[x]$, which we know is an Euclidean Domain
- Assume we can compute the GCD of two elements in R
- **Input:** $f, g \in R[x]$
- **Output:** $\gcd(f, g) \in R[x]$

GCD via field of fractions

- We can now compute the GCD over $R[x]$ as via it's field of fractions $\mathbb{F}[x]$, which we know is an Euclidean Domain
- Assume we can compute the GCD of two elements in R
- **Input:** $f, g \in R[x]$
- **Output:** $\gcd(f, g) \in R[x]$
- Algorithm:
  1. Compute $\text{content}(f), \text{content}(g), \text{pp}(f), \text{pp}(g)$

GCD via field of fractions

- We can now compute the GCD over $R[x]$ as via it's field of fractions $\mathbb{F}[x]$, which we know is an Euclidean Domain
- Assume we can compute the GCD of two elements in R
- **Input:** $f, g \in R[x]$
- **Output:** $\gcd(f, g) \in R[x]$
- Algorithm:
  1. Compute $\text{content}(f), \text{content}(g), \text{pp}(f), \text{pp}(g)$
  2. Let $h = \gcd(\text{content}(f), \text{content}(g))$ via algorithm for R

GCD via field of fractions

- We can now compute the GCD over $R[x]$ as via it's field of fractions $\mathbb{F}[x]$, which we know is an Euclidean Domain
- Assume we can compute the GCD of two elements in R
- **Input:** $f, g \in R[x]$
- **Output:** $\gcd(f, g) \in R[x]$
- Algorithm:
  1. Compute $\mathrm{content}(f), \mathrm{content}(g), \mathrm{pp}(f), \mathrm{pp}(g)$
  2. Let $h = \gcd(\mathrm{content}(f), \mathrm{content}(g))$ via algorithm for R
  3. Compute the monic GCD between $\dfrac{\mathrm{pp}(f)}{LC(f)}$ and $\dfrac{\mathrm{pp}(g)}{LC(g)}$ over $\mathbb{F}[x]$ – call it $q(x) \in \mathbb{F}[x]$

GCD via field of fractions

- We can now compute the GCD over $R[x]$ as via it's field of fractions $\mathbb{F}[x]$, which we know is an Euclidean Domain
- Assume we can compute the GCD of two elements in R
- **Input:** $f, g \in R[x]$
- **Output:** $\gcd(f, g) \in R[x]$
- Algorithm:
  1. Compute $\text{content}(f), \text{content}(g), \text{pp}(f), \text{pp}(g)$
  2. Let $h = \gcd(\text{content}(f), \text{content}(g))$ via algorithm for R
  3. Compute the monic GCD between $\dfrac{\text{pp}(f)}{LC(f)}$ and $\dfrac{\text{pp}(f)}{LC(f)}$ over $\mathbb{F}[x]$ – call it $q(x) \in \mathbb{F}[x]$
  4. Compute $b = \gcd(LC(f), LC(g))$ via algorithm for R

GCD via field of fractions

- We can now compute the GCD over $R[x]$ as via it's field of fractions $\mathbb{F}[x]$, which we know is an Euclidean Domain
- Assume we can compute the GCD of two elements in R
- **Input:** $f, g \in R[x]$
- **Output:** $\gcd(f, g) \in R[x]$
- Algorithm:
  1. Compute $\text{content}(f), \text{content}(g), \text{pp}(f), \text{pp}(g)$
  2. Let $h = \gcd(\text{content}(f), \text{content}(g))$ via algorithm for R
  3. Compute the monic GCD between $\dfrac{\text{pp}(f)}{LC(f)}$ and $\dfrac{\text{pp}(g)}{LC(g)}$ over $\mathbb{F}[x]$ – call it $q(x) \in \mathbb{F}[x]$
  4. Compute $b = \gcd(LC(f), LC(g))$ via algorithm for R
  5. Let $p = \text{pp}(b \cdot q) \in R[x]$

GCD via field of fractions

- We can now compute the GCD over $R[x]$ as via it's field of fractions $\mathbb{F}[x]$, which we know is an Euclidean Domain
- Assume we can compute the GCD of two elements in R
- **Input:** $f, g \in R[x]$
- **Output:** $\gcd(f, g) \in R[x]$
- Algorithm:
  1. Compute $\text{content}(f), \text{content}(g), \text{pp}(f), \text{pp}(g)$
  2. Let $h = \gcd(\text{content}(f), \text{content}(g))$ via algorithm for R
  3. Compute the monic GCD between $\dfrac{\text{pp}(f)}{LC(f)}$ and $\dfrac{\text{pp}(f)}{LC(f)}$ over $\mathbb{F}[x]$ – call it $q(x) \in \mathbb{F}[x]$
  4. Compute $b = \gcd(LC(f), LC(g))$ via algorithm for R
  5. Let $p = \text{pp}(b \cdot q) \in R[x]$
  6. Return $h \cdot p \in R[x]$

Example

- $f(x) = 6x^3 - 42x^2 + 72x - 60$ and $g(x) = 2x^2 - 6x - 20$

content $(f) = 6$ content $(g) = 2$

$pp(f) = x^3 - 7x^2 + 12x - 10$ $pp(g) = x^2 - 3x - 10$

$h = \gcd(\text{content}(f), \text{content}(g)) = 2$

$q = \gcd(x^3 - 7x^2 + 12x - 10, x^2 - 3x - 10) = x - 5$

$x^3 - 7x^2 + 12x - 10 = (x^2 - 3x - 10)(x - 4) + 10(x - 5)$

$x^2 - 3x - 10 = 10(x - 5) \cdot \frac{1}{10}(x + 2) + 0$

$P = 1 \cdot (x - 5)$

$h = 2$

$\gcd(f, g) = 2x - 10$

Deeper Look at GCD

- One disadvantage of the previous algorithm: bit complexity of intermediate numbers can be high

  Can we develop another algorithm that works over the ring itself?

want to decrease
bit complexity
(want somehow avoid dealing
with fractions)

Deeper Look at GCD

- One disadvantage of the previous algorithm: bit complexity of intermediate numbers can be high

  Can we develop another algorithm that works over the ring itself?
- Before we do that, let's look at the GCD over $\mathbb{F}[x]$ (an *Euclidean domain*) from an algebraic perspective:

$$\gcd(f(x), g(x)) = 1 \;\Leftrightarrow$$
$$\exists\ s(x), t(x) \in \mathbb{F}[x] \text{ s.t. } s(x) \cdot f(x) + t(x) \cdot g(x) = 1$$

Deeper Look at GCD

- One disadvantage of the previous algorithm: bit complexity of intermediate numbers can be high

  Can we develop another algorithm that works over the ring itself?
- Before we do that, let's look at the GCD over $\mathbb{F}[x]$ (an *Euclidean domain*) from an algebraic perspective:

$$\gcd(f(x), g(x)) = 1 \iff$$
$$\exists\ s(x), t(x) \in \mathbb{F}[x] \text{ s.t. } \boxed{s(x) \cdot f(x) + t(x) \cdot g(x) = 1}$$

- We can also assume w.l.o.g. that $\deg(s) < \deg(g)$ and $\deg(t) < \deg(f)$.
- Viewing the equation $s(x) \cdot f(x) + t(x) \cdot g(x) = 1$ as a linear system, we have:

  over $\mathbb{F}$

$$s_0 \cdot f_0 + t_0 \cdot g_0 = 1 \qquad \text{constant coefficient}$$

$$\sum_{i=0}^{k} s_i \cdot f_{k-i} + t_i \cdot g_{k-i} = 0 \qquad \text{coefficient of degree } k > 0$$

coeff. of $s(x)f(x) + tg$ of deg. k

Sylvester Matrix & Resultant

- In matrix form (for simplicity $\deg(f) = 3, \deg(g) = 2$):

$sf + tg = 1$

$$\begin{pmatrix} f_0 & 0 & g_0 & 0 & 0 \\ f_1 & f_0 & g_1 & g_0 & 0 \\ f_2 & f_1 & g_2 & g_1 & g_0 \\ f_3 & f_2 & 0 & g_2 & g_1 \\ 0 & f_3 & 0 & 0 & g_2 \end{pmatrix} \cdot \begin{pmatrix} s_0 \\ s_1 \\ t_0 \\ t_1 \\ t_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$f(x) = f_0 + f_1 x + f_2 x^2 + f_3 x^3$

$g(x) = g_0 + g_1 x + g_2 x^2$

$\deg(s) < \deg(g)$ and $\deg(t) < \deg(f)$

$s(x) = s_0 + s_1 x$

$t(x) = t_0 + t_1 x + t_2 x^2$

$f_2 \cdot s_0 + f_1 s_1$
$+ t_0 \cdot g_2 + t_1 g_1 +$
$t_2 g_0$

Sylvester Matrix & Resultant

- In matrix form (for simplicity $\deg(f) = 3, \deg(g) = 2$):

$sf + tg = 1$

$sf + tg = x^i$

$i \geq 0$

$$\underline{\begin{pmatrix} f_0 & 0 & g_0 & 0 & 0 \\ f_1 & f_0 & g_1 & g_0 & 0 \\ f_2 & f_1 & g_2 & g_1 & g_0 \\ f_3 & f_2 & 0 & g_2 & g_1 \\ 0 & f_3 & 0 & 0 & g_2 \end{pmatrix}} \cdot \begin{pmatrix} s_0 \\ s_1 \\ t_0 \\ t_1 \\ t_2 \end{pmatrix} = \begin{array}{c} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \end{array} \begin{array}{l} x^0 \\ x^1 \\ x^2 \\ \vdots \\ \end{array}$$

Definition (Sylvester Matrix)

The matrix arising from the linear system is called *Sylvester Matrix*. It is denoted by

$$Syl_x(f, g)$$

$e_1, e_2, e_3 \; \cdots, e_{d_f+d_g}$

$1 \quad x \quad x^2 \; \cdots \quad x^{d_f+d_g-1}$

Sylvester Matrix & Resultant

- In matrix form (for simplicity $\deg(f) = 3, \deg(g) = 2$):

$$\begin{pmatrix} f_0 & 0 & g_0 & 0 & 0 \\ f_1 & f_0 & g_1 & g_0 & 0 \\ f_2 & f_1 & g_2 & g_1 & g_0 \\ f_3 & f_2 & 0 & g_2 & g_1 \\ 0 & f_3 & 0 & 0 & g_2 \end{pmatrix} \cdot \begin{pmatrix} s_0 \\ s_1 \\ t_0 \\ t_1 \\ t_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Definition (Sylvester Matrix)

The matrix arising from the linear system is called *Sylvester Matrix*. It is denoted by

$$Syl_x(f, g)$$

Definition (Resultant)

The *Resultant* of f, g is the determinant of the Sylvester Matrix:

$$\mathrm{Res}_x(f, g) = \det(Syl_x(f, g))$$

Sylvester Matrix - General Case

$$f(x) = f_0 + f_1 x + \cdots + f_m x^m$$

$$g(x) = g_0 + g_1 x + \cdots + g_n x^n$$

$$\mathrm{Syl}_x(f,g) = \begin{pmatrix}
f_0 & 0 & & g_0 & & \\
f_1 & f_0 & \ddots & g_1 & \ddots & \\
\vdots & f_1 & f_0 & \vdots & \ddots & g_0 \\
f_m & \vdots & & g_n & & \vdots \\
 & f_m & \vdots & & \ddots & \\
 & & f_m & & & g_n
\end{pmatrix}$$

$$\underbrace{\qquad\qquad\qquad}_{n \text{ times}} \quad \underbrace{\qquad\qquad\qquad}_{m \text{ times}}$$

$(n+m) \times (n+m)$

Resultants - Properties

- Resultant between two polynomials f, g is an *algebraic invariant*, and it is very important in computational algebra and algebraic geometry[1]
- An important property is that the resultant is a *polynomial* over the *coefficients of* f, g

[1] As we will frequently see later in the course

Resultants - Properties

- Resultant between two polynomials f, g is an *algebraic invariant*, and it is very important in computational algebra and algebraic geometry[1]
- An important property is that the resultant is a *polynomial* over the *coefficients of* f, g
- From previous slides, another property is:

$$\mathrm{Res}_x(f, g) \neq 0 \;\Leftrightarrow\; \gcd(f, g) = 1 \quad \text{over } \mathbb{F}[x]$$

[1]As we will frequently see later in the course

Resultants - Properties

- Resultant between two polynomials f, g is an *algebraic invariant*, and it is very important in computational algebra and algebraic geometry[1]
- An important property is that the resultant is a *polynomial* over the *coefficients of* f, g
- From previous slides, another property is:

$$\mathrm{Res}_x(f,g) \neq 0 \;\Leftrightarrow\; \gcd(f,g) = 1 \quad \text{over } \mathbb{F}[x]$$

- The resultant can be defined over $R[x]$, since we didn't use any divisions!

$$\mathrm{Res}_x(f,g) = \det\begin{pmatrix} & \end{pmatrix} \leftarrow R^{(m+n)\times(m+n)}$$

[1] As we will frequently see later in the course

Resultants - Properties

- Resultant between two polynomials f, g is an *algebraic invariant*, and it is very important in computational algebra and algebraic geometry[1]
- An important property is that the resultant is a *polynomial* over the *coefficients of* f, g
- From previous slides, another property is:

$$\mathrm{Res}_x(f,g) \neq 0 \;\Leftrightarrow\; \gcd(f,g) = 1 \quad \text{over } \mathbb{F}[x]$$

- The resultant can be defined over $R[x]$, since we didn't use any divisions!
- Extending the property above, we have:

$$\mathrm{Res}_x(f,g) \neq 0 \;\Leftrightarrow\; \gcd(f,g) \in R \setminus \{0\} \quad \text{over } R[x]$$

In particular, f, g have no common polynomial factors over $R[x]$!

[1] As we will frequently see later in the course

Discriminant

- A particular case which you have seen before is the discriminant.

Discriminant

- A particular case which you have seen before is the discriminant.
- From calculus, we know that $f(x) \in \mathbb{R}[x]$ has a *double root* $\alpha \in \mathbb{R}$ iff α is a root of $f(x)$ and of $f'(x)$

Discriminant

- A particular case which you have seen before is the discriminant.
- From calculus, we know that $f(x) \in \mathbb{R}[x]$ has a *double root* $\alpha \in \mathbb{R}$ iff α is a root of $f(x)$ and of $f'(x)$
- That is, the polynomials $f(x)$ and $f'(x)$ have a common root.

Discriminant

- A particular case which you have seen before is the discriminant.
- From calculus, we know that $f(x) \in \mathbb{R}[x]$ has a *double root* $\alpha \in \mathbb{R}$ iff α is a root of $f(x)$ and of $f'(x)$
- That is, the polynomials $f(x)$ and $f'(x)$ have a common root.
- This implies that $\underline{x - \alpha} \mid \underline{\gcd(f, f')}$

gcd(f, f') is not $\underline{1}$

Discriminant

- A particular case which you have seen before is the discriminant.
- From calculus, we know that $f(x) \in \mathbb{R}[x]$ has a *double root* $\alpha \in \mathbb{R}$ iff α is a root of $f(x)$ and of $f'(x)$
- That is, the polynomials $f(x)$ and $f'(x)$ have a common root.
- This implies that $x - \alpha \mid \gcd(f, f')$
- By the properties of the resultant, we have

$$\mathrm{Res}_x(f, f') = 0$$

Discriminant

- A particular case which you have seen before is the discriminant.
- From calculus, we know that $f(x) \in \mathbb{R}[x]$ has a *double root* $\alpha \in \mathbb{R}$ iff α is a root of $f(x)$ and of $f'(x)$
- That is, the polynomials $f(x)$ and $f'(x)$ have a common root.
- This implies that $x - \alpha \mid \gcd(f, f')$
- By the properties of the resultant, we have

$$\mathrm{Res}_x(f, f') = 0$$

$$\boxed{f' = \frac{d}{dx} f}$$

- The *discriminant* of $f(x) \in R[x]$ is given by

$$\mathrm{disc}_x(f) := \mathrm{Res}_x(f, f')$$

Discriminant

- A particular case which you have seen before is the discriminant.
- From calculus, we know that $f(x) \in \mathbb{R}[x]$ has a *double root* $\alpha \in \mathbb{R}$ iff α is a root of $f(x)$ and of $f'(x)$
- That is, the polynomials $f(x)$ and $f'(x)$ have a common root.
- This implies that $x - \alpha \mid \gcd(f, f')$
- By the properties of the resultant, we have

$$\mathrm{Res}_x(f, f') = 0$$

$$\frac{-b \pm \sqrt{b^2-4ac}}{2a}$$

- The *discriminant* of $f(x) \in R[x]$ is given by

$$\mathrm{disc}_x(f) := \mathrm{Res}_x(f, f')$$

- Why is it called discriminant? If $f(x) = ax^2 + bx + c$, we get

(have double root)

$$\mathrm{disc}_x(f) = \underline{-a}_{\neq 0} \cdot (b^2 - 4ac)$$

Does this look familiar? :)

$\mathrm{disc}_x(f) = 0$ iff f is perfect square

- Resultants & Discriminants
- Modular GCD algorithm in $\mathbb{Z}[x]$
- Conclusion
- Acknowledgements

Using Resultant to Compute GCD

- Now that we know about the resultant of two polynomials, we can use it to devise modular algorithms in $\mathbb{Z}[x]$

Using Resultant to Compute GCD

- Now that we know about the resultant of two polynomials, we can use it to devise modular algorithms in $\mathbb{Z}[x]$
- We know how to compute GCDs over Euclidean domains $\mathbb{F}[x]$.

Using Resultant to Compute GCD

- Now that we know about the resultant of two polynomials, we can use it to devise modular algorithms in $\mathbb{Z}[x]$
- We know how to compute GCDs over Euclidean domains $\mathbb{F}[x]$.
- **Idea:** we can compute the GCD of f, g modulo a special prime p and from this GCD (over $\mathbb{Z}_p[x]$) to obtain $\gcd(f, g)$ over $\mathbb{Z}[x]$

Using Resultant to Compute GCD

- Now that we know about the resultant of two polynomials, we can use it to devise modular algorithms in $\mathbb{Z}[x]$
- We know how to compute GCDs over Euclidean domains $\mathbb{F}[x]$.
- **Idea:** we can compute the GCD of f, g modulo a special prime p and from this GCD (over $\mathbb{Z}_p[x]$) to obtain $\gcd(f, g)$ over $\mathbb{Z}[x]$
- Will any prime do?

$$f(x) = \overbrace{3x^3 - x^2 + 3x - 1}^{(3x-1)(x^2+1)} \text{ and } g(x) = \overbrace{3x^2 + 5x - 2}^{(3x-1)(x+2)}$$

$$h(x) := \gcd(f, g) = 3x - 1$$

Let's see how our idea will work out...

Example

- $f(x) = 3x^3 - x^2 + 3x - 1$ and $g(x) = 3x^2 + 5x - 2$

$$h(x) := \gcd(f, g) = 3x - 1$$

Example

- $f(x) = 3x^3 - x^2 + 3x - 1$ and $g(x) = 3x^2 + 5x - 2$

$$h(x) := \gcd(f, g) = 3x - 1$$

- $p = 3$ degree too small

$-f(x) = +x^2 + 1$ $\quad -g(x) = +x + 2$

$x^2 + 1 = (x+2)(x-2) + 1$

$\gcd_{\mathbb{Z}_3[x]}(f, g) = 1$

Example

- $f(x) = 3x^3 - x^2 + 3x - 1$ and $g(x) = 3x^2 + 5x - 2$

$$h(x) := \gcd(f, g) = \underline{3x - 1}$$

- $p = 3$ degree too small
- $p = 5$ degree too large

$$f(x) = 3x^3 - x^2 + 3x - 1$$

$$g(x) = 3x^2 - 2 = 3(x^2+1)$$

$$\gcd_{\mathbb{Z}_5[x]}(f, g) = x^2 + 1$$

$$3x^3 - x^2 + 3x - 1 = 3(x^2+1)\tfrac{1}{3}(3x-1) + 0$$

$$\begin{array}{l} 3x^3 - x^2 + 3x - 1 \\ \underline{3x^3 \qquad + 3x} \\ \quad -x^2 - 1 \end{array}$$

Example

- $f(x) = 3x^3 - x^2 + 3x - 1$ and $g(x) = 3x^2 + 5x - 2$

$$h(x) := \gcd(f, g) = 3x - 1$$

- $p = 3$ degree too small
- $p = 5$ degree too large
- $p = 7$ degree is good

Example

- $f(x) = 3x^3 - x^2 + 3x - 1$ and $g(x) = 3x^2 + 5x - 2$

$$h(x) := \gcd(f, g) = 3x - 1$$

- $p = 3$ degree too small
- $p = 5$ degree too large
- $p = 7$ degree is good
- What makes a prime bad?

Example

- $f(x) = 3x^3 - x^2 + 3x - 1$ and $g(x) = 3x^2 + 5x - 2$
$$h(x) := \gcd(f, g) = 3x - 1$$
- $p = 3$ degree too small
- $p = 5$ degree too large
- $p = 7$ degree is good
- What makes a prime bad?
- 3 is bad because it *decreases the degree* of *both* f, g

$h = \gcd(f, g)$

$LC(h) = \gcd(LC(f), LC(g))$

Example

$(3x-1)(x^2+1)$ $(3x-1)(x+2)$

- $f(x) = 3x^3 - x^2 + 3x - 1$ and $g(x) = 3x^2 + 5x - 2$

$$h(x) := \gcd(f, g) = 3x - 1$$

- $p = 3$ degree too small
- $p = 5$ ← divides $\mathrm{Res}_x(f/h, g/h)$ degree too large
- $p = 7$ degree is good
- What makes a prime bad? extra factors in gcd
- 3 is bad because it *decreases the degree* of *both* f, g
- Let's take a look at $\mathrm{Res}_x(f/h, g/h)$ $\underbrace{\mathrm{Res}_x(x^2+1, x+2)}$

$$\det\begin{pmatrix} 1 & 2 & 0 \\ 0 & 1 & 2 \\ 1 & 0 & 1 \end{pmatrix} = 1 + 4 = 5$$

not coprime modulo 5

($\mathrm{Res} \equiv 0 \bmod 5$)

Example

- $f(x) = 3x^3 - x^2 + 3x - 1$ and $g(x) = 3x^2 + 5x - 2$

$$h(x) := \gcd(f, g) = 3x - 1$$

- $p = 3$ degree too small
- $p = 5$ degree too large
- $p = 7$ degree is good
- What makes a prime bad?
- 3 is bad because it *decreases the degree* of *both* f, g
- Let's take a look at $\text{Res}_x(f/h, g/h)$
- Are these the only bad primes? YES!
  If p is a prime which *does not divide* $b = \gcd(LC(f), LC(g))$, then:
  1. $LC(h) \mid b$ $p \nmid LC(h)$
  2. $\deg(\gcd_{\mathbb{Z}_p[x]}(f, g)) \geq \deg(h)$
  3. p does not divide $\text{Res}_x(f, g) \Leftrightarrow \deg(\gcd_{\mathbb{Z}_p[x]}(f, g)) = \deg(h)$

Proof

- If p is a prime which *does not divide* $b = \gcd(LC(f), LC(g))$, then:
  1. $\cancel{LC(h)} \mid b$ $\quad p \nmid LC(h)$

$$LC(h) \mid \gcd(LC(f), LC(g)) = b$$

$$\Rightarrow p \nmid LC(h)$$

Proof

- If p is a prime which *does not divide* $b = \gcd(LC(f), LC(g))$, then:
  1. $LC(h) \mid b$
  2. $\deg(\gcd_{\mathbb{Z}_p[x]}(f, g)) \geq \deg(h)$

$$\deg(h) = \deg_{\mathbb{Z}_p[x]}(h)$$

$$h \mid f, g \Rightarrow h \mid f, g \text{ over } \mathbb{Z}_p[x]$$

Proof

- If p is a prime which *does not divide* $b = \gcd(LC(f), LC(g))$, then:
  1. $LC(h) \mid b$
  2. $\deg(\gcd_{\mathbb{Z}_p[x]}(f, g)) \geq \deg(h)$
  3. p does not divide $\mathrm{Res}_x(\frac{f}{h}, \frac{g}{h}) \Leftrightarrow \deg(\gcd_{\mathbb{Z}_p[x]}(f, g)) = \deg(h)$

$\Downarrow$

$\frac{f}{h}, \frac{g}{h}$ they are coprime

over $\mathbb{Z}_p[x]$ $\Rightarrow$ $\mathrm{Res}_x\left[\frac{f}{h}, \frac{g}{h}\right] \neq 0$

over $\mathbb{Z}_p$

$s\frac{f}{h} + t\frac{g}{h} = 1 \iff sf + tg = h$

What is the size of output?

- Now that we have seen that the resultant is closely related to GCD and its modular versions, let's see how we can use it to bound the complexity of the GCD

given $f, g \in \mathbb{Z}[x]$

what is the complexity

of coefficients of h?

What is the size of output?

- Now that we have seen that the resultant is closely related to GCD and its modular versions, let's see how we can use it to bound the complexity of the GCD
- Given a polynomial $f(x) \in \mathbb{Z}[x]$, $f(x) = f_0 + f_1 x + \cdots + f_d x^d$, we consider two norms:
  1. The *height* of f is the magnitude of its largest coefficient:
     $$\|f\|_\infty = \max_{0 \le k \le d} |f_k|$$
  2. The ℓ_1 norm of $f(x)$ is:
     $$\|f\|_1 = \sum_{k=0}^{d} |f_d|$$

sum of absolute values of all coeffs.

What is the size of output?

- Now that we have seen that the resultant is closely related to GCD and its modular versions, let's see how we can use it to bound the complexity of the GCD
- Given a polynomial $f(x) \in \mathbb{Z}[x]$, $f(x) = f_0 + f_1 x + \cdots + f_d x^d$, we consider two norms:
  1. The *height* of f is the magnitude of its largest coefficient:
  $$\|f\|_\infty = \max_{0 \leq k \leq d} |f_k|$$
  2. The ℓ_1 norm of $f(x)$ is:
  $$\|f\|_1 = \sum_{k=0}^{d} |f_d|$$

Proposition (Coefficient Bound on Factors)

Given $f(x), g(x), h(x) \in \mathbb{Z}[x]$ *such that* $f = gh$ *and* $\deg(f) = d$, *we have:*

1. $\|h\|_\infty \leq (d+1)^{1/2} \cdot 2^d \cdot \|f\|_\infty$
2. $\|h\|_\infty \cdot \|g\|_\infty \leq \|h\|_1 \cdot \|g\|_1 \leq (d+1)^{1/2} \cdot 2^d \cdot \|f\|_\infty$

Bounding Bad Primes

- Let $A = \max(\|f\|_\infty, \|g\|_\infty)$ and $d = \deg(f) \geq \deg(g)$
- Bad primes are the ones which divide $\gcd(LT(f), LT(g))$ or divide $\mathrm{Res}_x(f/h, g/h)$. How to bound their complexity?

Bounding Bad Primes

- Let $A = \max(\|f\|_\infty, \|g\|_\infty)$ and $d = \deg(f) \geq \deg(g)$
- Bad primes are the ones which divide $\gcd(LT(f), LT(g))$ or divide $\mathrm{Res}_x(f/h, g/h)$. How to bound their complexity?
- We already know that $LC(f) \leq \|f\|_\infty \leq A$ and $LC(g) \leq \|g\|_\infty \leq A$. How to bound the absolute value of $\mathrm{Res}_x(f/h, g/h)$?

Bounding Bad Primes

- Let $A = \max(\|f\|_\infty, \|g\|_\infty)$ and $d = \deg(f) \geq \deg(g)$
- Bad primes are the ones which divide $\gcd(LT(f), LT(g))$ or divide $\mathrm{Res}_x(f/h, g/h)$. How to bound their complexity?
- We already know that $LC(f) \leq \|f\|_\infty \leq A$ and $LC(g) \leq \|g\|_\infty \leq A$. How to bound the absolute value of $\mathrm{Res}_x(f/h, g/h)$?
- We know that $\mathrm{Res}_x(f/h, g/h)$ is the determinant of the Sylvester matrix of f/h and g/h,
  1. By lemma from previous slide, $\|f/h\|_\infty, \|g/h\|_\infty \leq (d+1)^{1/2} \cdot 2^d A$

Bounding Bad Primes

- Let $A = \max(\|f\|_\infty, \|g\|_\infty)$ and $d = \deg(f) \geq \deg(g)$
- Bad primes are the ones which divide $\gcd(LT(f), LT(g))$ or divide $\mathrm{Res}_x(f/h, g/h)$. How to bound their complexity?
- We already know that $LC(f) \leq \|f\|_\infty \leq A$ and $LC(g) \leq \|g\|_\infty \leq A$. How to bound the absolute value of $\mathrm{Res}_x(f/h, g/h)$?
- We know that $\mathrm{Res}_x(f/h, g/h)$ is the determinant of the Sylvester matrix of f/h and g/h,
  1. By lemma from previous slide, $\|f/h\|_\infty, \|g/h\|_\infty \leq (d+1)^{1/2} \cdot 2^d A$
  2. Thus, $\mathrm{Res}_x(f/h, g/h)$ is a determinant of a "$2d \times 2d$ matrix" with entries bounded by $(d+1)^{1/2} \cdot 2^d A$

$\det(M)$

$(d+1)^{1/2}\, 2^d A$

Bounding Bad Primes

- Let $A = \max(\|f\|_\infty, \|g\|_\infty)$ and $d = \deg(f) \geq \deg(g)$
- Bad primes are the ones which divide $\gcd(LT(f), LT(g))$ or divide $\text{Res}_x(f/h, g/h)$. How to bound their complexity?
- We already know that $LC(f) \leq \|f\|_\infty \leq A$ and $LC(g) \leq \|g\|_\infty \leq A$. How to bound the absolute value of $\text{Res}_x(f/h, g/h)$?
- We know that $\text{Res}_x(f/h, g/h)$ is the determinant of the Sylvester matrix of f/h and g/h,
  1. By lemma from previous slide, $\|f/h\|_\infty, \|g/h\|_\infty \leq (d+1)^{1/2} \cdot 2^d A$
  2. Thus, $\text{Res}_x(f/h, g/h)$ is a determinant of a "$2d \times 2d$ matrix" with entries bounded by $(d+1)^{1/2} \cdot 2^d A$
  3. So can bound $|\text{Res}_x(f/h, g/h)|$ by the straightforward bound:

$$|\text{Res}_x(f/h, g/h)| \leq (2d)! \cdot [(d+1)^{1/2} \cdot 2^d A]^{2d}$$

$$|\det(M)| = \left|\sum_{\sigma \in S_{2d}} (-1)^\sigma \cdot \underbrace{\prod M_{i\sigma(i)}}\right| \leq (2d)! \cdot [(d+1)^{1/2} \cdot 2^d A]^{2d}$$

Algorithm

- **Input**: primitive polynomials $f, g \in \mathbb{Z}[x]$
- **Output:** $h = \gcd(f, g)$

Algorithm

- **Input**: primitive polynomials $f, g \in \mathbb{Z}[x]$
- **Output:** $h = \gcd(f, g)$
- Algorithm:
  1. Compute $b = \gcd(LT(f), LT(g))$, and set $B \in \mathbb{N}$

$$B = (2d)! \left[(d+1)^{1/2} \, 2^d A \right]^{2d} \cdot c$$

constant

$$B > \left| \mathrm{Res}_x(f/h, g/h) \right|, \ b$$

Algorithm

- **Input**: primitive polynomials $f, g \in \mathbb{Z}[x]$
- **Output:** $h = \gcd(f, g)$
- Algorithm:
  1. Compute $b = \gcd(LT(f), LT(g))$, and set $B \in \mathbb{N}$
  2. Pick random prime $p \in [2B, 4B]$
  3. Compute $p(x) = \gcd_{\mathbb{Z}_p[x]}(f, g)$

$\deg(p) = \deg(h)$

Algorithm

- **Input**: primitive polynomials $f, g \in \mathbb{Z}[x]$
- **Output:** $h = \gcd(f, g)$
- Algorithm:
  1. Compute $b = \gcd(LT(f), LT(g))$, and set $B \in \mathbb{N}$
  2. Pick random prime $p \in [2B, 4B]$
  3. Compute $p(x) = \gcd_{\mathbb{Z}_p[x]}(f, g)$
  4. Compute $q, f^*, g^* \in \mathbb{Z}[x]$ with height $\underline{< p/2}$ satisfying:

$$q \equiv bp(x) \mod p, \quad f^* \cdot q \equiv b \cdot f \mod p, \quad g^* \cdot q \equiv b \cdot q \mod p$$

Algorithm

$3x-1$

$\cancel{\boxed{3x+6}}$

- **Input**: primitive polynomials $f, g \in \mathbb{Z}[x]$
- **Output:** $h = \gcd(f, g)$
- Algorithm:
  1. Compute $b = \gcd(LT(f), LT(g))$, and set $B \in \mathbb{N}$
  2. Pick random prime $p \in [2B, 4B]$
  3. Compute $p(x) = \gcd_{\mathbb{Z}_p[x]}(f, g)$
  4. Compute $q, f^*, g^* \in \mathbb{Z}[x]$ with height $< p/2$ satisfying:

$$q \equiv bp \mod p, \quad f^* \cdot q \equiv b \cdot f \mod p, \quad g^* \cdot q \equiv b \cdot q \mod p$$

  5. If

$$\underline{\|f^*\|_1 \cdot \|q\|_1 \leq B} \quad \text{and} \quad \underline{\|g^*\|_1 \cdot \|q\|_1 \leq B}$$

Return q.
Otherwise go back to step 2.

$q \in \mathbb{Z}_p[x]$ actually

Algorithm

- **Input**: primitive polynomials $f, g \in \mathbb{Z}[x]$
- **Output:** $h = \gcd(f, g)$
- Algorithm:
  1. Compute $b = \gcd(LT(f), LT(g))$, and set $B \in \mathbb{N}$
  2. Pick random prime $p \in [2B, 4B]$
  3. Compute $p(x) = \gcd_{\mathbb{Z}_p[x]}(f, g)$
  4. Compute $q, f^*, g^* \in \mathbb{Z}[x]$ with height $< p/2$ satisfying:

  $$q \equiv bp \mod p, \quad f^* \cdot q \equiv b \cdot f \mod p, \quad g^* \cdot q \equiv b \cdot q \mod p$$

  5. If

  $$\|f^*\|_1 \cdot \|q\|_1 \leq B \quad \text{and} \quad \|g^*\|_1 \cdot \|q\|_1 \leq B$$

  Return q.
  Otherwise go back to step 2.
- Correctness follows by previous slides, and probability the our random prime does not work is $\leq 1/2$.

- Resultants & Discriminants
- Modular GCD algorithm in $\mathbb{Z}[x]$
- Conclusion
- Acknowledgements

Conclusion

In today's lecture, we learned

- Resultants, Discriminants and their properties
  1. Capture whether two polynomials have common factor
  2. Capture complexity of coefficients in $\gcd(f, g)$
  3. Capture whether polynomial has multiple factors
  4. Much more to be seen!
- How to use the resultant to design and analyze a modular gcd algorithm

Acknowledgement

Based largely on

- Arne's notes

  https://cs.uwaterloo.ca/~r5olivei/courses/2021-winter-cs487/lec7-ref.pdf
- Lectures 3 and 4 from Madhu's notes http://people.csail.mit.edu/madhu/FT98/