CS 4424

GCD, XGCD

eschost@uwo.ca

GCD of polynomials

First definition

- Let A and B be in $k[x]$.

  $k[x]$ is the ring of polynomials with coefficients in k

- A Greatest Common Divisor of A and B is a polynomial G such that
  - G divides A
  - G divides B
  - if C divides both A and B, it divides G.

- If G and H are GCD's of A and B, then $G = \ell H$, for some constant $\ell \neq 0$.

- So usually we say that THE GCD is the one with leading coefficient $=1$.

  A monic polynomial is a polynomial with leading coefficient $=1$.

GCD of polynomials

Irreducible polynomials

- Let A be in $k[x]$. Then A is irreducible if it cannot be factored into $A = PQ$.

  (except if either P or Q is a constant)

- Examples in $\mathbb{Q}[x]$

$$x, \quad 2x, \quad x^2+1, \quad x^2+2, \quad x^4+20, \quad x^3+324x-2342.$$

  But $x^2 - 3x + 2$ is not irreducible.

- Any polynomial can be uniquely factored into a product

$$A = \alpha P_1^{e_1} \cdots P_s^{e_s}$$

  with α constant, P_i irreducible, monic and $e_i \geq 1$.

GCD of polynomials

Second definition of the GCD

- Let A and B be in $k[x]$. Factor A and B as

$$A = \alpha Q_1^{e_1} \cdots Q_s^{e_s}$$

and

$$B = \beta P_1^{f_1} \cdots P_r^{f_r}.$$

- Example: $A = x(x^2+1)$ and $B = (x-1)(x+1)(x^2+1)^2$ gives

$$s = 2, \quad Q_1 = x, e_1 = 1, \quad Q_2 = (x^2+1), e_2 = 1$$

and

$$r = 3, \quad P_1 = x-1, f_1 = 1, \quad P_2 = x+1, f_2 = 1, \quad P_3 = x^2+1, f_3 = 2.$$

GCD of polynomials

Second definition of the GCD

- Let $R_1, \ldots, R_t$ be the common irreducible factors between A and B.
- For any R_i, let g_i be the minimum of the exponents it has in A and B.
- Then $\gcd(A, B) = R_1^{g_1} \cdots R_t^{g_t}$.
- Example:

$$t = 1, \quad R_1 = x^2 + 1, \quad g_1 = 1,$$

so $\gcd(A, B) = x^2 + 1$.

The fact that these two definitions are equivalent requires a proof, that I'm not going to do.

Algorithms

Facts

- The previous definitions do not lead to an easy algorithm.
- To do better: Euclid's algorithm.

Complexity

- The naive version of Euclid's algorithm takes $O(n^2)$ for polynomials of degree n.
- The fast version takes $O(\mathsf{M}(n)\log(n))$.

A few useful rules

Prop.

- $\gcd(A, B) = \gcd(B, A)$.

  The definition is symmetric.

- $\gcd(A, 0) = A/\text{leading coefficient}(A)$.

  A divides A, and A divides 0, so A divides their GCD. Conversely, the GCD divides A. So the GCD is a constant times A.

- $\gcd(A, c) = 1$ if c is a non-zero constant.

  Any polynomial that divides c is a constant.

The main idea

Prop.

- For all A, B in $k[x]$,

$$\gcd(A, B) = \gcd(A, B \bmod A) = \gcd(B, A \bmod B).$$

Proof.

- Let $R = B \bmod A$. Then

$$R = B - AQ.$$

- Let $G = \gcd(A, B)$ and $H = \gcd(A, R)$.
- G divides A and B, so G divides R.

  Property of the GCD for H: G divides H.
- H divides A and R, so H divides B.

  Property of the GCD for G: H divides G.

Euclid's algorithm

gcd(A, B)

- if $\deg(A) < \deg(B)$ then return $\gcd(B, A)$.

  so now we assume that $\deg(A) \geq \deg(B)$

- if $B = 0$ then return A/leading coefficient(A).

  second rule

- return $\gcd(B, A \bmod B)$

  previous slide

Towards the iterative presentation

Setup.

- We rewrite $A_0 = A$, $A_1 = B$.
- We assume $\deg(A_0) \geq \deg(A_1)$ (otherwise, swap them).

Steps.

- $\gcd(A_0, A_1) = \gcd(A_1, A_2)$ $\qquad A_2 = A_0 \bmod A_1$
- $\gcd(A_1, A_2) = \gcd(A_2, A_3)$ $\qquad A_3 = A_1 \bmod A_2$
- $\ldots$
- $\gcd(A_i, A_{i+1}) = \gcd(A_{i+1}, A_{i+2})$ $\qquad A_{i+2} = A_i \bmod A_{i+1}$
- $\ldots$
- $\gcd(A_N, 0) = A_N/\text{leading coefficient}(A_N)$.

The iterative presentation

Setup.

- We rewrite $A_0 = A$, $A_1 = B$.
- We assume $\deg(A_0) \geq \deg(A_1)$ (otherwise, swap them).

Steps.

- $i = 1$
- while $A_i \neq 0$
- $\quad A_{i+1} = A_{i-1} \bmod A_i$
- $\quad i$++
- return A_{i-1}/leading coefficient(A_{i-1})

Complexity

Setup.

- $n = \deg(A_0)$
- then, all polynomials have degree $\leq n$.

Naive analysis.

- We do at most $n + 1$ Euclidean divisions.
- Euclidean division in degree $\leq n$ takes $O(n^2)$ operations.
- So the total cost is $O(n^3)$.

Correct result, but we can do much better.

A more careful analysis of Euclidean division

Prop.

- If $\deg(A) = n$ and $\deg(B) = m$, we can compute the quotient and remainder of A by B in at most

$$2(n-m)(n+1)$$

operations.

Proof.

- We do $n - m$ reduction steps.
- Each takes $\leq 2(n+1)$ operations.

A better analysis of Euclid's gcd algorithm

Prop.

- The total cost is $O(n^2)$.

Proof. Let $n_i = \deg(A_i)$ be the degrees of the successive remainders.

- Then the cost of computing A_{i+1} is at most

$$2(n_{i-1} - n_i)(n_{i-1} + 1) \leq 2(n_{i-1} - n_i)(n + 1).$$

- So the total cost is at most

$$\sum_{i=1}^{N-1} 2(n_{i-1} - n_i)(n + 1) \leq 2(n + 1) \sum_{i=1}^{N-1} (n_{i-1} - n_i)$$

- The sum simplifies into $n_0 - n_{N-1} \leq n$
- So the total cost is at most $2(n + 1)n = O(n^2)$.

Extended gcd

Prop.

- Given A and B, one can compute $G = \gcd(A, B)$, as well as Bézout coefficients U, V such that

$$AU + BV = G, \quad \deg(U) < \deg(B), \quad \deg(V) < \deg(A)$$

  by a small modification of Euclid's algorithm.

Special case.

- We say that A and B are coprime if $\gcd(A, B) = 1$.
- In that case the Bézout coefficients satisfy

$$AU + BV = 1.$$

Example: complex numbers

How to compute with complex numbers

- complex multiplication is multiplication modulo $x^2 + 1$;
- complex inversion is extended gcd with $x^2 + 1$.
  - suppose $z = a + ib$
  - compute $G = \gcd(a + xb, x^2 + 1)$ and the coefficients $U(x), V(x)$
  - facts: $G = 1$, $\deg(U) < 2$ and $\deg(V) < 1$
  - then $(u_0 + u_1 x)(a + bx) + v_0(x^2 + 1) = 1$
  - evaluating at $x = i$ gives $(u_0 + u_1 i)(a + bi) = 1$

More general example (optional)

Suppose that P in $k[x]$ is irreducible: it has no divisor, other than constants or itself. Then for A in $k[x]$:

- either P divides A, and then $\gcd(A, P) = P$
- or $\gcd(A, P) = 1$.

Remember how we defined $k[x]/P$ as

- the set of all polynomials of degree less than $\deg(P)$
- with addition and multiplication defined modulo P.

Now we also have inversion modulo P:

- for $A \neq 0$ in $k[x]/P$, $\gcd(A, P) = 1$
- so there exists U, V with $AU + PV = 1$ (as polynomials)
- so $AU = 1$ in $k[x]/P$.

Towards the extended Euclidean algorithm

Getting the quotients.

- replace the step

$$A_{i+1} = A_{i-1} \bmod A_i$$

by

$$Q_i = A_{i-1} \operatorname{div} A_i$$

and

$$A_{i+1} = A_{i-1} - Q_i A_i$$

- remark that we still have

$$A_{i+1} = A_{i-1} \bmod A_i$$

- the algorithm is still $O(n^2)$

The extended Euclidean algorithm

Additionnally to (A_i), we also compute sequence (U_i) and (V_i) with

$$U_0 = 1, \quad U_1 = 0, \quad U_{i+1} = U_{i-1} - Q_i U_i$$

and

$$V_0 = 0, \quad V_1 = 1, \quad V_{i+1} = V_{i-1} - Q_i V_i$$

Prop.

- For $0 \leq i \leq N$, we have

$$A_0 U_i + A_1 V_i = A_i$$

Proof.

- By induction ($i = 0$ and 1 initiate the induction).

Prop.

- For $i = N$ (when we get the gcd), we have

$$A_0 U_N + A_1 V_N = A_N.$$

Degrees and complexity

Roughly speaking

- the degrees of the U_i and V_i increase;
- the degrees of the A_i decrease.

Precisely

- $\deg(U_i) = \deg(Q_2) + \cdots + \deg(Q_{i-1})$ $\quad i \geq 2$
- $\deg(V_i) = \deg(Q_1) + \cdots + \deg(Q_{i-1})$ $\quad i \geq 2$

But $\deg(Q_i) = \deg(A_{i-1}) - \deg(A_i)$ so

- $\deg(U_i) = \deg(A_1) - \deg(A_{i-1}) \leq n$ $\quad i \geq 2$
- $\deg(V_i) = \deg(A_0) - \deg(A_{i-1}) \leq n$ $\quad i \geq 2$

Consequence: the complexity is still $O(n^2)$.

Rational reconstruction

With Newton iteration, we can expand

$$S(x) = \frac{N(x)}{D(x)} = s_0 + s_1 x + s_2 x^2 + \cdots$$

Assuming you know sufficiently many terms, it is possible to go backwards and recover $N(x)/D(x)$.

Prop.

- This is a problem of linear algebra, so it can be solved in theory.
- Euclid's algorithm give a better algorithm.

  When we get to fast Euclidean algorithm, this will be almost optimal.

Sketch of the algorithm

Suppose that:

- we know that $\deg(N) \leq n$ and $\deg(D) \leq d$;
- we know $s_0, \ldots, s_{n+d}$.

We run the extended Euclidean algorithm with input $A_0 = x^{n+d+1}$ and $A_1 = G = s_0 + \cdots + s_{n+d}x^{n+d}$.

- For $i = 0$, let $U_0 = 1, V_0 = 0, A_0 = x^{n+d+1}$.
- For $i = 1$, let $U_1 = 0, V_1 = 1, A_1 = G$.
- For $i \geq 1$
  - $Q_i = A_{i-1} \text{ div } A_i$
  - $A_{i+1} = A_{i-1} - Q_i A_i$,
  - $U_{i+1} = U_{i-1} - Q_i U_i$,
  - $V_{i+1} = V_{i-1} - Q_i V_i$.

Recovering N/D

At each step, we maintain the invariant $U_i x^{n+d+1} + V_i G = A_i$.

Moreover:

- the degrees of the A_i decrease;
- the degrees of the V_i increase.

Prop.

- Let i be the first index with $\deg(A_i) \leq n$.
- Then $\deg(V_i) = n + d + 1 - \deg(A_{i-1}) \leq d$.
- Hence, $A_i/V_i = N/D$.

IQ test

Problem: find the next term.

$U:\ 1,1,1,1,1,1,1,1$

$V:\ 0,1,1,2,3,5,8,13$

$W:\ 12,134,222,21,-3898,-40039,-347154,-2929918,-24657854$

Answer: 1, 21 and -207605083.

How? The sequences U, V, W satisfy linear recurrences with constant coefficients:

$$U_{n+1} = U_n,$$

$$V_{n+2} = V_{n+1} + V_n,$$

$$W_{n+4} = 12W_{n+3} - 33W_{n+2} + 22W_{n+1} + 19W_n.$$

Generating series

Given a sequence $u = u_0, u_1, \ldots,$ we can construct the series

$$S = \sum_{i \geq 0} u_i x^i.$$

This is the generating series of u.

- The properties of u (recurrence) translate to properties of S.

Simple case

- $u_n = 2^n$ (equivalently, $u_0 = 1$ and $u_{n+1} - 2u_n = 0$)
- generating series

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

Rational series

The series of the previous example is rational.

Prop.

- The generating series S is rational:

$$S = \frac{N(x)}{D(x)},$$

with

$$D(x) = 1 + a_{k-1}x + \cdots + a_1 x^{k-1} + a_0 x^k \quad \text{and} \quad \deg(N) < \deg(D)$$

if and only if the sequence u satisfies the recurrence

$$u_{n+k} + a_{k-1}u_{n+k-1} + \cdots + a_1 u_{n+1} + a_0 u_n = 0$$

rational series $\iff$ recurrence with constant coefficients

Proof on an example

Let's check for recurrences of order 2, with

$$u_0 = \alpha, \quad u_1 = \beta, \quad u_{n+2} + au_{n+1} + bu_n = 0$$

and

$$S = \sum_{i \geq 0} u_i x^i.$$

1. Multiply the recurrence relation by x^{n+2}:

$$u_{n+2}x^{n+2} + au_{n+1}x^{n+2} + bu_n x^{n+2} = 0.$$

2. Sum, for $n \geq 0$:

$$S - (\alpha + \beta x) + ax(S - \alpha) + bx^2 S = 0.$$

3. Rearrange

$$S = \frac{\alpha + (\beta + \alpha a)x}{1 + ax + bx^2}.$$

Consequence

Suppose that you know that a sequence s_i satisfies a recurrence of order k:

- then, the generating series is rational with numerator of degree $n < k$ and denominator of degree $d = k$
- you need $s_0, \ldots, s_{n+d}$, so up to s_{2k-1}.
- you apply the Extended Euclidean Algorithm
- you get the first i with $\deg(A_i) \leq k - 1$.

Matrices in Euclid's algorithm

Notation as before:

- $A_0, A_1, \ldots$ the successives remainders
- $Q_1, Q_2, \ldots$ the quotients.

We can write the transformation $(A_{i-1}, A_i) \to (A_i, A_{i+1})$ in a matrix way:

$$\begin{bmatrix} A_i \\ A_{i+1} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -Q_i \end{bmatrix} \begin{bmatrix} A_{i-1} \\ A_i \end{bmatrix}.$$

Multiplying matrices, we see that for all i, we can write

$$\begin{bmatrix} A_i \\ A_{i+1} \end{bmatrix} = R_i \begin{bmatrix} A_0 \\ A_1 \end{bmatrix}.$$

HGCD

Main idea

- to compute the (x)gcd, it is too costly to compute all remainders;
- we are going to do big steps by skipping a lot of them.

Half-gcd

- we suppose $\deg(A_0) > \deg(A_1)$
- $n = \deg(A_0)$
- there exists a unique j such that

$$\deg(A_j) \geq \frac{n}{2} > \deg(A_{j+1})$$

- the half-gcd algorithm computes the matrix R_j.

The GCD matrix

The GCD matrix is the matrix that corresponds to

$$\begin{bmatrix} A_N \\ 0 \end{bmatrix} = R_N \begin{bmatrix} A_0 \\ A_1 \end{bmatrix}.$$

If we find it, we can get:

- the GCD A_N;
- the Bézout coefficients (first row).

HGCD $\rightarrow$ GCD

Recursive algorithm for computing the GCD matrix.

GCD_matrix(A_0, A_1)

- $S_0 = \mathsf{HGCD}(A_0, A_1)$
- Compute A_j and A_{j+1}
- If $A_{j+1} = 0$, return S_0
- Compute Q_{j+1} and

$$S_1 = \begin{bmatrix} 0 & 1 \\ 1 & -Q_{j+1} \end{bmatrix}$$

- Compute A_{j+2}
- If $A_{j+2} = 0$, return $S_1 S_0$
- Compute $S_2 = \mathsf{GCD\_matrix}(A_{j+1}, A_{j+2})$ and return $S_2 S_1 S_0$

Cost analysis

Notation

- Let G(n) be the cost of GCD_matrix in degree n
- Let H(n) be the cost of HGCD in degree n.

  Fact: $\mathsf{H}(n) = O(\mathsf{M}(n)\log(n))$.
- Recall that quotient and remainder take $O(\mathsf{M}(n))$.

Recurrence

$$\mathsf{G}(n) = \mathsf{G}(n/2) + O(\mathsf{M}(n)\log(n))$$

Solving it gives

$$\mathsf{G}(n) = O(\mathsf{M}(n)\log(n))$$

Main idea of the HGCD

In Euclidean division

- when you divide two polynomials (of high degree),
- the remainder does depend on all coefficients
- but the quotient depends only on the high-degree ones.

You can see it:

- in the slow algorithm, you construct Q using the high-degree terms only
- in the fast algorithm, you construct Q by a truncated series product.

HGCD (for nice polynomials)

Assume

$$\deg(A_0) = n, \quad \deg(A_1) = n-1, \quad \ldots \quad \deg(A_i) = n-i$$

so all quotients have degree 1.

Consequence

- the half-gcd matrix of A_0, A_1 has degrees about $n/2$ (up to ± 1)

More generally

- the matrix of R_j to the remainder of degree $n-j$ has degree about j (up to ± 1)

Divide-and-conquer

Given

$$A_0 = a_n X^n + a_{n-1} X^{n-1} + \cdots, \quad A_1 = a'_{n-1} X^{n-1} + a'_{n-2} X^{n-2} + \cdots$$

we let

$$B_0 = a_n X^{n/2} + a_{n-1} X^{n/2-1} + \cdots, \quad B_1 = a'_{n-1} X^{n/2-1} + a'_{n-2} X^{n/2-2} + \cdots$$

and we compute

$$S_0 = \mathsf{HGCD}(B_0, B_1)$$

If our polynomials are nice

- the degrees of S_0 should be about $n/4$
- applying it to (A_0, A_1) should give remainders of degree $n - n/4 = 3n/4$.

Continuing the divide-and-conquer

Let A'_0, A'_1 be obtained by

$$\begin{bmatrix} A'_0 \\ A'_1 \end{bmatrix} = S_0 \begin{bmatrix} A_0 \\ A_1 \end{bmatrix}.$$

These are the remainders of degree $(3n/4, 3n/4 - 1)$.

So they look like

$$A'_0 = \alpha_{3n/4} X^{3n/4} + \alpha_{3n/4-1} X^{3n/4-1} + \cdots$$

$$A'_1 = \alpha'_{3n/4-1} X^{3n/4-1} + \alpha'_{3n/4-2} X^{3n/4-2} + \cdots$$

Continuing the divide-and-conquer

We define

$$B'_0 = \alpha_{3n/4} X^{n/2} + \alpha_{3n/4-1} X^{n/2-1} + \cdots$$

$$B'_1 = \alpha'_{3n/4-1} X^{n/2-1} + \alpha'_{3n/4-2} X^{n/2-2} + \cdots$$

and we compute

$$S_1 = \mathsf{HGCD}(B'_0, B'_1)$$

If our polynomials are nice

- the degrees of S_1 should be about $n/4$
- applying it to (A'_0, A'_1) should give remainders of degree $3n/4 - n/4 = n/2$.

Summary: HGCD algorithm

HGCD(A_0, A_1)

- $S_0 = \mathsf{HGCD}(B_0, B_1)$ $\qquad B_i = A_i \text{ div } X^{n/2}$
- Compute A_j and A_{j+1}
- If $A_{j+1} = 0$, return S_0
- Compute Q_{j+1} and

$$S_1 = \begin{bmatrix} 0 & 1 \\ 1 & -Q_{j+1} \end{bmatrix}$$

- Compute A_{j+2}
- If $A_{j+2} = 0$, return $S_1 S_0$
- Compute $S_2 = \mathsf{HGCD}(B_{j+1}, B_{j+2})$ $\qquad B_i = A_i \text{ div } X^{n/2}$
- Return $S_2 S_1 S_0$

Complexity

The algorithm does:

- 2 recursives calls in degree $n/2$
- 1 Euclidean division in degree $\leq n$
- some products of 2×2 matrices in degree $\leq n$

Recurrence

$$\mathsf{H}(n) = 2\mathsf{H}(n/2) + O(\mathsf{M}(n))$$

Solving it gives

$$H(n) \in O(\mathsf{M}(n)\log(n)).$$