Lecture 5: Univariate Polynomial Division & Newton Iteration

Rafael Oliveira

University of Waterloo
Cheriton School of Computer Science

rafael.oliveira.teaching@gmail.com

January 24, 2021

Overview

- Formal Power Series Ring & Reversal
- Newton Iteration & Inversion
- Division via Newton Iteration
- Conclusion
- Acknowledgements

Dividing Polynomials

$d_a := \deg(a)$
$d_b := \deg(b)$

- In Lecture 1, we saw how to divide polynomials over $\mathbb{Z}[x]$
- Running time $O(d_a d_b)$ to compute $a = \underline{q} \cdot b + \underline{r}$

assumed $b(x)$ was <u>monic</u>

(leading monomial is a unit
$\therefore$ leading monomial is 1)

$b(x) = \underline{1 \cdot x^2} + x + 2$ is monic

$c(x) = \underline{3x^3} - x + \underline{1}$ is <u>not</u> monic

not unit in $\mathbb{Z}$.

Dividing Polynomials

- In Lecture 1, we saw how to divide polynomials over $\mathbb{Z}[x]$
- Running time $O(d_a d_b)$ to compute $a = q \cdot b + r$
- That algorithm (Euclidean division) generalizes to the setting of $\mathbb{F}[x]$, where $\mathbb{F}$ is a field.

(over $\mathbb{F}[x]$ every polynomial is monic

because every nonzero element of $\mathbb{F}$ is a unit)

Dividing Polynomials

- In Lecture 1, we saw how to divide polynomials over $\mathbb{Z}[x]$
- Running time $O(d_a d_b)$ to compute $a = q \cdot b + r$ $O(d^2)$
- That algorithm (Euclidean division) generalizes to the setting of $\mathbb{F}[x]$, where $\mathbb{F}$ is a field.
- Also saw in previous lecture how to multiply two polynomials of degree $O(d)$ in $O(d \log d)$ time

$d_a, d_b = O(d)$

Dividing Polynomials

- In Lecture 1, we saw how to divide polynomials over $\mathbb{Z}[x]$
- Running time $O(d_a d_b)$ to compute $a = q \cdot b + r$
- That algorithm (Euclidean division) generalizes to the setting of $\mathbb{F}[x]$, where $\mathbb{F}$ is a field.
- Also saw in previous lecture how to multiply two polynomials of degree $O(d)$ in $O(d \log d)$ time
- Is division with remainder more complex than multiplication?
  1. Can compute division with remainder with $O(d \log d)$ operations in $\mathbb{F}$?
  2. Can we use fast multiplication to speedup division?

Dividing Polynomials

- In Lecture 1, we saw how to divide polynomials over $\mathbb{Z}[x]$
- Running time $O(d_a d_b)$ to compute $a = q \cdot b + r$
- That algorithm (Euclidean division) generalizes to the setting of $\mathbb{F}[x]$, where $\mathbb{F}$ is a field.
- Also saw in previous lecture how to multiply two polynomials of degree $O(d)$ in $O(d \log d)$ time
- Is division with remainder more complex than multiplication?
  1. Can compute division with remainder with $O(d \log d)$ operations in $\mathbb{F}$?
  2. Can we use fast multiplication to speedup division?
- YES!

Dividing Polynomials

- In Lecture 1, we saw how to divide polynomials over $\mathbb{Z}[x]$
- Running time $O(d_a d_b)$ to compute $a = q \cdot b + r$
- That algorithm (Euclidean division) generalizes to the setting of $\mathbb{F}[x]$, where $\mathbb{F}$ is a field.
- Also saw in previous lecture how to multiply two polynomials of degree $O(d)$ in $O(d \log d)$ time
- Is division with remainder more complex than multiplication?
    1. Can compute division with remainder with $O(d \log d)$ operations in $\mathbb{F}$?
    2. Can we use fast multiplication to speedup division?
- YES!
- in 70s, Borodin and Moenck; Strassen; Sieveking and Kung; derived a division algorithm with $O(d \log^2 d \log \log d)$ operations

Ring of Formal Power Series

- We have seen the polynomial ring $\mathbb{F}[x]$, whose elements are of the form
$$p(x) = \underbrace{p_0 + p_1 x + \cdots + p_d x^d}_{\text{finite sum}}$$

$p_i \in \mathbb{F}$

Ring of Formal Power Series

- We have seen the polynomial ring $\mathbb{F}[x]$, whose elements are of the form
$$p(x) = p_0 + p_1 x + \cdots + p_d x^d$$
- We can extend this ring to the *formal power series ring* $\mathbb{F}[[x]]$, whose elements are now:

$$p(x) = p_0 + p_1 x + p_2 x^2 + \cdots$$

$$(p_k)_{k \geq 0} \longleftrightarrow p_0 + p_1 x + p_2 x^2 + \cdots + p_d x^d + \cdots$$

$$\mathbb{F}[x] \subset \mathbb{F}[[x]]$$

$\uparrow$
sequences with finitely many non-zero elements

Ring of Formal Power Series

- We have seen the polynomial ring $\mathbb{F}[x]$, whose elements are of the form
$$p(x) = p_0 + p_1 x + \cdots + p_d x^d$$
- We can extend this ring to the *formal power series ring* $\mathbb{F}[[x]]$, whose elements are now:
$$p(x) = p_0 + p_1 x + p_2 x^2 + \cdots$$
- Addition and multiplication done similar to ring of polynomials

$$\underbrace{(p+q)_k = p_k + q_k}_{\text{component-wise addition}} \qquad \underbrace{(pq)_k = \sum_{i=0}^{k} p_i q_{k-i}}_{\text{polynomial multiplication}}$$

Ring of Formal Power Series

- We have seen the polynomial ring $\mathbb{F}[x]$, whose elements are of the form
$$p(x) = p_0 + p_1 x + \cdots + p_d x^d$$
- We can extend this ring to the *formal power series ring* $\mathbb{F}[[x]]$, whose elements are now:
$$p(x) = p_0 + p_1 x + p_2 x^2 + \cdots$$
- Addition and multiplication done similar to ring of polynomials
$$(p+q)_k = p_k + q_k \qquad (pq)_k = \sum_{i=0}^{k} p_i q_{k-i}$$
- We don't care about convergence of power series, as we will not evaluate them.

$P(x) = 1 + x + x^2 + \cdots$

$\cancel{P(1) = \infty}$ not defined

Property of Power Series Rings

- Now more elements have inverses in the ring:
- $p(x) = p_0 + p_1 x + \cdots \in \mathbb{F}[[x]]$ has an inverse in $\mathbb{F}[[x]]$ iff $p_0 \neq 0$.

Constant term non zero

$$\frac{1}{1-x} = \underbrace{1 + x + x^2 + \cdots}_{\in \mathbb{F}[[x]]}$$

$1 - x \in \mathbb{F}[[x]]$

$p \cdot q = 1 + 0 \cdot x +$

$p \cdot q = 1$

$q(x) = q_0 + q_1 x + \cdots$ defined by

$q_0 = p_0^{-1}$ (exists because $p_0 \neq 0$ in $\mathbb{F}$)

$0 = q_0 \cdot p_1 + q_1 p_0 \rightarrow \boxed{q_1 = -\frac{q_0 p_1}{p_0} = -q_0^2 p_1}$

$0 = q_0 \cdot p_k + q_1 \cdot p_{k-1} + \cdots + \boxed{q_k} \cdot p_0 \quad q_k = -q_0 (q_0 p_k + \cdots + q_{k-1} p_1)$

Reversal of Polynomials

- Given polynomial $p(x) = p_0 + p_1 x + \cdots + p_k x^k$ we can algebraically *reverse* coefficients of $p(x)$ getting

$$q(x) = p_k + p_{k-1} x + \cdots + p_0 x^k$$

Computationally this is easy

Reversal of Polynomials

- Given polynomial $p(x) = p_0 + p_1 x + \cdots + p_k x^k$ we can algebraically *reverse* coefficients of $p(x)$ getting

$$q(x) = p_k + p_{k-1}x + \cdots + p_0 x^k$$

- Operation is called *reversal*, can be done algebraically as follows:

$$rev_k(p) := q(x) = x^k \cdot p(1/x)$$

$$p(x) = 3x^2 + x + 2$$

$$p(1/x) = \frac{3}{x^2} + \frac{1}{x} + 2$$

$$x^2 \cdot p(1/x) = 3 + x + 2x^2 = rev_2(p)$$

$$x^3 \cdot p(1/x) = 3x + x^2 + 2x^3 = x \cdot rev_2(p)$$

Reversal of Polynomials

- Given polynomial $p(x) = p_0 + p_1 x + \cdots + p_k x^k$ we can algebraically *reverse* coefficients of $p(x)$ getting

$$q(x) = p_k + p_{k-1} x + \cdots + p_0 x^k$$

- Operation is called *reversal*, can be done algebraically as follows:

$$rev_k(p) := q(x) = x^k \cdot p(1/x)$$

- Note that if $k \geq \deg(p)$ then the reversal just has extra factor of x^{k-d}

Reversal of Polynomials

- Given polynomial $p(x) = p_0 + p_1 x + \cdots + p_k x^k$ we can algebraically *reverse* coefficients of $p(x)$ getting

$$q(x) = p_k + p_{k-1}x + \cdots + p_0 x^k$$

- Operation is called *reversal*, can be done algebraically as follows:

$$rev_k(p) := q(x) = x^k \cdot p(1/x)$$

- Note that if $k \geq \deg(p)$ then the reversal just has extra factor of x^{k-d}
- When a polynomial $p(x)$ of degree d is *monic* (i.e., leading coefficient 1), we have $rev_d(p)$ is *invertible* over $\mathbb{F}[[x]]$

$$rev_d(p) = \underbrace{p_d}_{1} + p_{d-1}x + \cdots + p_0 \cdot x^d$$

$\Rightarrow rev_d(p)$ is invertible over $\mathbb{F}[[x]]$.

Newton Iteration

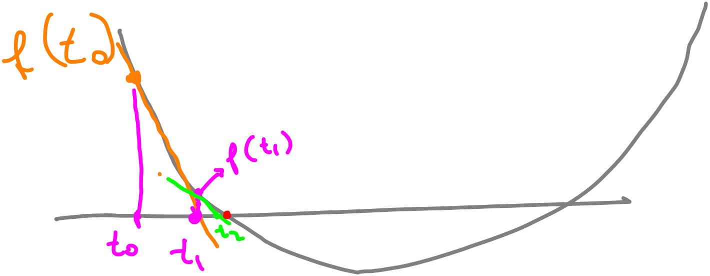
- *Newton iteration:* given differentiable $f : \mathbb{R} \to \mathbb{R}$, compute successive approximations to solutions of $f(t) = 0$ (finding roots of f)

Newton Iteration

- *Newton iteration:* given differentiable $f : \mathbb{R} \to \mathbb{R}$, compute successive approximations to solutions of $f(t) = 0$ (finding roots of f)
- From initial approximation t_0, get next approximation by

$$\underline{t_{i+1}} = \underline{t_i} - \frac{f(t_i)}{f'(t_i)}$$

next guess



Newton Iteration

- *Newton iteration:* given differentiable $f : \mathbb{R} \to \mathbb{R}$, compute successive approximations to solutions of $f(t) = 0$ (finding roots of f)
- From initial approximation t_0, get next approximation by

$$t_{i+1} = t_i - \frac{f(t_i)}{f'(t_i)}$$

can be defined over $\mathbb{F}[[x]]$

- Can use this to find inverse of a polynomial $p(x)$ over the ring $\mathbb{F}[[x]]$:
  1. Function to "find root of:"

$$\Phi(y) = \frac{1}{y} - p(x)$$

so long as f is differentiable

$$\Phi(p^{-1}) = \frac{1}{p^{-1}} - p = p - p = 0$$

p^{-1} is root of $\Phi(y)$

Newton Iteration

- *Newton iteration:* given differentiable $f : \mathbb{R} \to \mathbb{R}$, compute successive approximations to solutions of $f(t) = 0$ (finding roots of f)
- From initial approximation t_0, get next approximation by

$$t_{i+1} = t_i - \frac{f(t_i)}{f'(t_i)}$$

- Can use this to find inverse of a polynomial $p(x)$ over the ring $\mathbb{F}[[x]]$:
  1. Function to "find root of:"

$$\Phi(y) = \frac{1}{y} - p(x)$$

  2. Derivative (over y):

$$\Phi'(y) = -\frac{1}{y^2}$$

$$\frac{d}{dy}\left(\frac{1}{y} - p(x)\right) = \frac{d}{dy}\left(\frac{1}{y}\right) = -y^{-2}$$

Newton Iteration

- *Newton iteration:* given differentiable $f : \mathbb{R} \to \mathbb{R}$, compute successive approximations to solutions of $f(t) = 0$ (finding roots of f)
- From initial approximation t_0, get next approximation by

$$t_{i+1} = t_i - \frac{f(t_i)}{f'(t_i)}$$

- Can use this to find inverse of a polynomial $p(x)$ over the ring $\mathbb{F}[[x]]$:
  1. Function to "find root of:"

$$\boxed{\Phi(y) = \frac{1}{y} - p(x)}$$

  2. Derivative (over y):

$$\Phi'(y) = -\frac{1}{y^2}$$

  3. Newton iteration step:

$$f_{i+1} = f_i - \frac{\frac{1}{f_i} - p}{-1/f_i^2} = 2f_i - pf_i^2$$

$$\frac{\frac{1}{f_i} - p(x)}{-\frac{1}{f_i^2}}$$

Newton Iteration

- Have from Newton Iteration:

$$f_{i+1} = f_i - \frac{\frac{1}{f_i} - p}{-1/f_i^2} = 2f_i - pf_i^2$$

Newton Iteration

- Have from Newton Iteration:

$$f_{i+1} = f_i - \frac{\frac{1}{f_i} - p}{-1/f_i^2} = 2f_i - pf_i^2$$

- How do we start finding the inverse of $p(x)$?
  1. First guess: $f_0 := p_0^{-1}$

makes the first coefficient of power series correct

$$p(x)^{-1} = \underline{f_0} + (\ldots x + x^2 + \ldots)$$

$$f_0 = p_0^{-1} \qquad p(x) \cdot p^{-1}(x) = 1 + 0 \cdot x + 0 x^2 + \ldots$$

$$f_0 \cdot p_0 = 1$$

Newton Iteration

- Have from Newton Iteration:

$$f_{i+1} = f_i - \frac{\frac{1}{f_i} - p}{-1/f_i^2} = 2f_i - pf_i^2$$

- How do we start finding the inverse of $p(x)$?
  1. First guess: $f_0 := p_0^{-1}$
  2. $f_0 \cdot p = 1 + \underline{p_1 p_0^{-1}} x + \cdots$ (right on constant term)

wrong

right !

Newton Iteration

- Have from Newton Iteration:

$$f_{i+1} = f_i - \frac{\frac{1}{f_i} - p}{-1/f_i^2} = 2f_i - pf_i^2$$

- How do we start finding the inverse of $p(x)$?
  1. First guess: $f_0 := p_0^{-1}$
  2. $f_0 \cdot p = 1 + p_1 p_0^{-1} x + \cdots$ (right on constant term)
  3. $f_1 = 2f_0 - f_0^2 \cdot p(x)$

$$f_1 = 2f_0 - f_0^2 (p_0 + p_1 x + p_2 x^2 + \cdots)$$

$$= \underline{f_0} - f_0^2 p_1 x - f_0^2 p_2 x^2 - f_0^2 (\cdots)$$

$$f_1 p = (f_0 p_0) + (f_0 \cdot p_1 + (-f_0^2 p_1) \cdot p_0) x + \cdots$$

$$= 1 + (f_0 p_1 - f_0 p_1) x + \cdots = 1 + 0 \cdot x + \text{other stuff}$$

Newton Iteration

- Have from Newton Iteration:

$$f_{i+1} = f_i - \frac{\frac{1}{f_i} - p}{-1/f_i^2} = 2f_i - pf_i^2$$

- How do we start finding the inverse of $p(x)$?
  1. First guess: $f_0 := p_0^{-1}$
  2. $f_0 \cdot p = 1 + p_1 p_0^{-1} x + \cdots$ (right on constant term)
  3. $f_1 = 2f_0 - f_0^2 \cdot p(x)$
  4. $$p \cdot f_1 = 2f_0 \cdot p - f_0^2 \cdot p^2 = 1 + 0 \cdot x - (p_1/p_0)^2 \cdot x^2 + \cdots$$

Newton Iteration

- Have from Newton Iteration:

$$f_{i+1} = f_i - \frac{\frac{1}{f_i} - p}{-1/f_i^2} = 2f_i - pf_i^2$$

- How do we start finding the inverse of $p(x)$?
  1. First guess: $f_0 := p_0^{-1}$
  2. $f_0 \cdot p = 1 + p_1 p_0^{-1} x + \cdots$ (right on constant term)
  3. $f_1 = 2f_0 - f_0^2 \cdot p(x)$
  4. $$p \cdot f_1 = 2f_0 \cdot p - f_0^2 \cdot p^2 = 1 + 0 \cdot x - (p_1/p_0)^2 \cdot x^2 + \cdots$$

Right up to linear term...

$$p \cdot f_1 \equiv 1 \mod x^2$$

$P \cdot f_1 = 1 + 0 \cdot x + \cancel{x^2 \cdot (\text{something})}$

formal power series

Newton Iteration Theorem

Theorem (Newton Iteration)

If $p(x) \in \mathbb{F}[x]$ is such that $p_0 = 1$ and $f_0 = 1, f_1, \ldots$ are the polynomials obtained by the Newton Iteration, then for all $i \geq 0$:

$$\underset{\uparrow}{p \cdot f_i} \equiv 1 \underbrace{\mod x^{2^i}}$$

f_i is the inverse of p up to precision x^{2^i}

i^{th} guess from Newton iteration

Newton Iteration Theorem

Theorem (Newton Iteration)

If $p(x) \in \mathbb{F}[x]$ is such that $p_0 = 1$ and $f_0 = 1, f_1, \ldots$ are the polynomials obtained by the Newton Iteration, then for all $i \geq 0$:

$$p \cdot f_i \equiv 1 \mod x^{2^i}$$

- Proof by induction: base case $i = 0$ we saw in previous slide

$$p \cdot f_0 = 1 \bmod x$$

$$\Longleftrightarrow$$

constant term of

$$p \cdot f_0 = 1$$

Newton Iteration Theorem

Theorem (Newton Iteration)

If $p(x) \in \mathbb{F}[x]$ is such that $p_0 = 1$ and $f_0 = 1, f_1, \ldots$ are the polynomials obtained by the Newton Iteration, then for all $i \geq 0$:

$$p \cdot f_i \equiv 1 \mod x^{2^i}$$

- Proof by induction: base case $i = 0$ we saw in previous slide
- Assume $p \cdot f_i \equiv 1 \mod x^{2^i}$: $(1 - p f_i)^2 = x^{2^{i+1}} (\text{polynomial})^2 \equiv 0 \mod x^{2^{i+1}}$

$$\begin{aligned}
\underline{1 - p \cdot f_{i+1}} &\equiv 1 - p \cdot (2f_i - pf_i^2) \mod x^{2^{i+1}} \\
&\equiv 1 - 2 \cdot p \cdot f_i + p^2 \cdot f_i^2 \mod x^{2^{i+1}} \quad \text{expand} \\
&\equiv (1 - p \cdot f_i)^2 \mod x^{2^{i+1}} \quad \text{perfect square} \\
&\equiv 0 \mod x^{2^{i+1}} \quad \text{induction}
\end{aligned}$$

$f_{i+1} = 2f_i - p f_i^2$

$1 - p \cdot f_i \equiv 0 \mod x^{2^i} \iff 1 - p f_i = x^{2^i} (\text{polynomial})$

Newton Iteration Algorithm for Polynomial Inversion

- **Input:** $p(x) \in \mathbb{F}[x]$ of degree d such that $p_0 = 1$, $t \in \mathbb{N}$
- **Output:** inverse $f_t(x) \in \mathbb{F}[x]$ of $p(x)$ up to degree 2^t. That is:

$$f_t(x) \cdot p(x) \equiv 1 \mod x^{2^t}$$

Newton Iteration Algorithm for Polynomial Inversion

- **Input:** $p(x) \in \mathbb{F}[x]$ of degree d such that $p_0 = 1$, $t \in \mathbb{N}$
- **Output:** inverse $f_t(x) \in \mathbb{F}[x]$ of $p(x)$ up to degree 2^t. That is:

$$f_t(x) \cdot p(x) \equiv 1 \mod x^{2^t}$$

- $f_0 = 1$

Newton Iteration Algorithm for Polynomial Inversion

- **Input:** $p(x) \in \mathbb{F}[x]$ of degree d such that $p_0 = 1$, $t \in \mathbb{N}$
- **Output:** inverse $f_t(x) \in \mathbb{F}[x]$ of $p(x)$ up to degree 2^t. That is:

$$f_t(x) \cdot p(x) \equiv 1 \mod x^{2^t}$$

- $f_0 = 1$
- For $i = 0, \ldots, t-1$:
  Compute $f_{i+1} = 2f_i - p \cdot f_i^2 \mod x^{2^i}$ Newton iteration
- Return f_t

previous slide gives us correctness of this algorithm

Newton Iteration Algorithm for Polynomial Inversion

- **Input:** $p(x) \in \mathbb{F}[x]$ of degree d such that $p_0 = 1$, $t \in \mathbb{N}$
- **Output:** inverse $f_t(x) \in \mathbb{F}[x]$ of $p(x)$ up to degree 2^t. That is:

$$f_t(x) \cdot p(x) \equiv 1 \mod x^{2^t}$$

- $f_0 = 1$

addition   squaring (multiplication)

- For $i = 0, \ldots, t-1$:
  Compute $f_{i+1} = 2f_i - p \cdot f_i^2 \mod x^{2^i}$ i^{th} step deg 2^i

  multiplication

- Return f_t

Assumptions on polynomial multiplication

1. $M(d) :=$ # field operations to multiply two degree $\leq d$ polynomials
2. $d \leq M(d)$ and $M(2d) \geq 2 \cdot M(d)$

Analysis

- The algorithm from previous slide runs in time $O(M(\underset{\text{precision}}{\underbrace{2^t}}))$

Analysis

- The algorithm from previous slide runs in time $O(M(2^t))$
- We have $f_{i+1} = 2f_i - \underline{p \cdot f_i^2} \mod x^{2^i}$
- # field operations to compute f_{i+1} from f_i is at most

$$2 \cdot \underline{M(2^i)} + 2 \cdot 2^i$$

as we perform all computations modulo x^{2^i}

Analysis

- The algorithm from previous slide runs in time $O(M(2^t))$
- We have $f_{i+1} = 2f_i - p \cdot f_i^2 \mod x^{2^i}$ ← t times
- # field operations to compute f_{i+1} from f_i is at most

$$2 \cdot M(2^i) + 2 \cdot 2^i$$

  as we perform all computations modulo x^{2^i}
- Total running time is:

$$\sum_{i=1}^{t} (2 \cdot M(2^i) + 2^{i+1})$$

Analysis

- The algorithm from previous slide runs in time $O(M(2^t))$
- We have $f_{i+1} = 2f_i - p \cdot f_i^2 \mod x^{2^i}$
- $\#$ field operations to compute f_{i+1} from f_i is at most

$$2 \cdot M(2^i) + 2 \cdot 2^i$$

  as we perform all computations modulo x^{2^i}
- Total running time is:

$$\sum_{i=1}^{t} (2 \cdot M(2^i) + 2^{i+1})$$

- Using $2 \cdot M(2^i) \le M(2^{i+1})$ and $2^i \le M(2^i)$, we get:

$\le \sum_{i=1}^{t} M(2^{i+1})$

$$\sum_{i=1}^{t} (2 \cdot M(2^i) + 2^{i+1}) = 2 \cdot \sum_{i=1}^{t} M(2^i) + \sum_{i=1}^{t} 2^{i+1}$$

$\le M(2^{i+1})$

$$\le \sum_{i=2}^{t+1} M(2^i) + \sum_{i=2}^{t+1} M(2^i) \le 4 \cdot M(2^{t+1})$$

$2 \cdot \sum M(2^i) \le M(2^{t+1}) \cdot (1 + \frac{1}{2} + \frac{1}{4} + \dots)$

~~Analysis~~ Punch line

We showed that we can invert element $p(x) \in \mathbb{F}[x]$ $p_0 = 1$ up to precision d in "same # operations" it takes to multiply two polynomials of degree $2d$.

Division With Remainder

- **Input:** polynomials $a(x), b(x) \in \mathbb{F}[x]$ $d = \deg(a) \geq \overset{d_b}{\deg(b)} \geq 0$
- **Output:** polynomials $q(x), r(x) \in \mathbb{F}[x]$ such that

$$a(x) = b(x) \cdot q(x) + r(x)$$

$\deg(r) < \deg(b)$

Division With Remainder

- **Input:** polynomials $a(x), b(x) \in \mathbb{F}[x]$ $d = \deg(a) \geq \deg(b) \geq 0$
- **Output:** polynomials $q(x), r(x) \in \mathbb{F}[x]$ such that

$$a(x) = b(x) \cdot q(x) + r(x)$$

- Assume $b(x)$ is *monic* (easy to do)

Division With Remainder

- **Input:** polynomials $a(x), b(x) \in \mathbb{F}[x]$ $d = \deg(a) \geq \deg(b) \geq 0$
- **Output:** polynomials $q(x), r(x) \in \mathbb{F}[x]$ such that

$$\underbrace{a(x) = b(x) \cdot q(x)}_{d_b \quad d-d_b} + r(x) \qquad d_r \leqslant d_b - 1$$

- Assume $b(x)$ is *monic* (easy to do)
- If $d_b = \deg(b)$, we have:

$$rev_d(r)$$

$$rev_d(a) = \underbrace{rev_{d-d_b}(q) \cdot rev_{d_b}(b)}_{} + \underline{x^{d-d_b+1} \cdot rev_{d_b-1}(r)}$$

$$rev_d(bq) = rev_{d_b}(b) \cdot rev_{d-d_b}(q)$$

$$\underbrace{x^d}_{x^{d_b} \cdot x^{d-d_b}} \cdot b(1/x) \cdot q(1/x)$$

$$rev_{d_b}(b) \cdot rev_{d-d_b}(q)$$

Division With Remainder

deg(q) = $d - d_b$

- **Input:** polynomials $a(x), b(x) \in \mathbb{F}[x]$ $d = \deg(a) \geq \deg(b) \geq 0$
- **Output:** polynomials $q(x), r(x) \in \mathbb{F}[x]$ such that

$$a(x) = b(x) \cdot q(x) + r(x)$$

- Assume $b(x)$ is *monic* $\Rightarrow$ $rev_{d_b}(b)$ has constant term $= 1$ $\Rightarrow$ b is invertible! (easy to do)
- If $d_b = \deg(b)$, we have:

$$rev_d(a) = rev_{d-d_b}(q) \cdot rev_{d_b}(b) + \underline{x^{d-d_b+1}} \cdot rev_{d_b-1}(r)$$

- Thus:

$$rev_d(a) \equiv rev_{d-d_b}(q) \cdot rev_{d_b}(b) \quad \underline{\bmod x^{d-d_b+1}}$$

invert $rev_{d_b}(b)$

$$\underline{rev_d(a) \cdot rev_{d_b}(b)^{-1} \equiv rev_{d-d_b}(q)} \quad \underline{\bmod x^{d-d_b+1}}$$

reversal of quotient q can be computed by a product of two polynomials we know!

Division With Remainder

- **Input:** polynomials $a(x), b(x) \in \mathbb{F}[x]$ $d = \deg(a) \geq \deg(b) \geq 0$
- **Output:** polynomials $q(x), r(x) \in \mathbb{F}[x]$ such that

$$a(x) = b(x) \cdot q(x) + r(x)$$

- Assume $b(x)$ is *monic* (easy to do)
- If $d_b = \deg(b)$, we have:

$$rev_d(a) = rev_{d-d_b}(q) \cdot rev_{d_b}(b) + x^{d-d_b+1} \cdot rev_{d_b-1}(r)$$

- Thus:

$$\begin{aligned} rev_d(a) &\equiv rev_{d-d_b}(q) \cdot rev_{d_b}(b) \mod x^{d-d_b+1} \\ rev_d(a) \cdot rev_{d_b}(b)^{-1} &\equiv \underline{rev_{d-d_b}(q)} \mod x^{d-d_b+1} \end{aligned}$$

- We get

$$q = rev_{d-d_b}(rev_{d-d_b}(q))$$

Division With Remainder

- **Input:** polynomials $a(x), b(x) \in \mathbb{F}[x]$ $d = \deg(a) \geq \deg(b) \geq 0$
- **Output:** polynomials $q(x), r(x) \in \mathbb{F}[x]$ such that

$$a(x) = b(x) \cdot q(x) + r(x)$$

- Assume $b(x)$ is *monic* (easy to do)
- If $d_b = \deg(b)$, we have:

$$rev_d(a) = rev_{d-d_b}(q) \cdot rev_{d_b}(b) + x^{d-d_b+1} \cdot rev_{d_b-1}(r)$$

- Thus:

$$\begin{aligned} rev_d(a) &\equiv rev_{d-d_b}(q) \cdot rev_{d_b}(b) \mod x^{d-d_b+1} \\ rev_d(a) \cdot rev_{d_b}(b)^{-1} &\equiv rev_{d-d_b}(q) \mod x^{d-d_b+1} \end{aligned}$$

- We get

$$q = rev_{d-d_b}(rev_{d-d_b}(q))$$

- And $r = a - b \cdot q$

Runtime and Analysis

- Correctness follows from properties of reversal

Runtime and Analysis

- Correctness follows from properties of reversal
- Running time follows from our algorithm for inversion and two more polynomial multiplication

$rev_{d_b}(b)$ $\quad O(M(2d))$

$rev_d(a) \cdot rev_{d_b}(b)$ $\quad O(M(2(d-d_b+1)))$

$q \leftarrow$ linear time

$r = a - b \cdot q$ $\quad O(M(d))$

$O(M(2d))$

Analysis

Conclusion

In today's lecture, we learned

- Properties of Ring of Power Series
- Newton iteration
- How to use Newton Iteration to compute inverses in ring of power series
- How to use reversal and Newton iteration to perform fast polynomial division with remainder
- Division with remainder in $O(d \log d)$ field operations

Acknowledgement

- Based largely on Arne's notes

  `https://cs.uwaterloo.ca/~r5olivei/courses/2021-winter-cs487/lec5-ref.pdf`