Lecture 3: Evaluation, Interpolation and Multiplication of Polynomials

Rafael Oliveira

University of Waterloo
Cheriton School of Computer Science

rafael.oliveira.teaching@gmail.com

January 17, 2021

Overview

- Polynomial Evaluation
- Polynomial Multiplication
- Polynomial Interpolation
- Conclusion
- Acknowledgements

- Polynomial Evaluation
- Polynomial Multiplication
- Polynomial Interpolation
- Conclusion
- Acknowledgements

Polynomial Evaluation

- **Setting:** ring $R[x]$, can perform basic operations $(+, \times)$ over R
- **Input:** elements $\alpha, a_0, \ldots, a_d \in R$
- **Output:** $p(\alpha)$, where

$$p(x) = a_0 + a_1 x + \cdots + a_d x^d \in R[x]$$

Polynomial Evaluation

- **Setting:** ring $R[x]$, can perform basic operations $(+, \times)$ over R
- **Input:** elements $\alpha, a_0, \ldots, a_d \in R$
- **Output:** $p(\alpha)$, where

$$p(x) = a_0 + a_1 x + \cdots + a_d x^d \in R[x]$$

- Naive algorithm:
    - Compute $\alpha^2, \alpha^3, \ldots, \alpha^d$ ($d - 1$ multiplications)

Polynomial Evaluation

- **Setting:** ring $R[x]$, can perform basic operations $(+, \times)$ over R
- **Input:** elements $\alpha, a_0, \ldots, a_d \in R$
- **Output:** $p(\alpha)$, where

$$p(x) = a_0 + a_1 x + \cdots + a_d x^d \in R[x]$$

- Naive algorithm:
  - Compute $\alpha^2, \alpha^3, \ldots, \alpha^d$ ($d - 1$ multiplications)
  - Compute $a_j \alpha^j$ (d multiplications)

Polynomial Evaluation

- **Setting:** ring $R[x]$, can perform basic operations $(+, \times)$ over R
- **Input:** elements $\alpha, a_0, \ldots, a_d \in R$
- **Output:** $p(\alpha)$, where

$$p(x) = a_0 + a_1 x + \cdots + a_d x^d \in R[x]$$

- Naive algorithm:
  - Compute $\alpha^2, \alpha^3, \ldots, \alpha^d$ ($d - 1$ multiplications)
  - Compute $a_j \alpha^j$ (d multiplications)
  - Add $a_j \alpha^j$ (d additions)

Polynomial Evaluation

- **Setting:** ring $R[x]$, can perform basic operations $(+, \times)$ over R
- **Input:** elements $\alpha, a_0, \ldots, a_d \in R$
- **Output:** $p(\alpha)$, where

$$p(x) = a_0 + a_1 x + \cdots + a_d x^d \in R[x]$$

- Naive algorithm:
  - Compute $\alpha^2, \alpha^3, \ldots, \alpha^d$ ($d-1$ multiplications)
  - Compute $a_j \alpha^j$ (d multiplications)
  - Add $a_j \alpha^j$ (d additions)

  $2d-1$ $\times$
  d $+$
- Can we do better?

Polynomial Evaluation

- **Setting:** ring $R[x]$, can perform basic operations $(+, \times)$ over R
- **Input:** elements $\alpha, a_0, \ldots, a_d \in R$
- **Output:** $p(\alpha)$, where

$$p(x) = a_0 + a_1 x + \cdots + a_d x^d \in R[x]$$

- Naive algorithm:
  - Compute $\alpha^2, \alpha^3, \ldots, \alpha^d$ ($d-1$ multiplications)
  - Compute $a_j \alpha^j$ (d multiplications)
  - Add $a_j \alpha^j$ (d additions)
- Can we do better?
- Horner's algorithm (a.k.a. Horner's rule):
  - Write

$$p(x) = (\cdots((a_d x + a_{d-1}) \cdot x + a_{d-2}) \cdot x + a_{d-3}) \cdot x + \cdots a_1) \cdot x + a_0$$

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

Polynomial Evaluation

- **Setting:** ring $R[x]$, can perform basic operations $(+, \times)$ over R
- **Input:** elements $\alpha, a_0, \ldots, a_d \in R$
- **Output:** $p(\alpha)$, where

$$p(x) = a_0 + a_1 x + \cdots + a_d x^d \in R[x]$$

- Naive algorithm:
  - Compute $\alpha^2, \alpha^3, \ldots, \alpha^d$ — $2d-1$ $\times$ — ($d-1$ multiplications)
  - Compute $a_j \alpha^j$ — d $+$ — (d multiplications)
  - Add $a_j \alpha^j$ — (d additions)
- Can we do better?
- Horner's algorithm (a.k.a. Horner's rule):
  - Write

$$p(x) = (\cdots((\overbrace{a_d x}^{d} \underbrace{+}_{d-1} a_{d-1}) \overbrace{\cdot x}^{d-1} \underbrace{+}_{d-2} a_{d-2}) \overbrace{\cdot x}^{d-2} + a_{d-3}) \cdot x + \cdots a_1) \overbrace{\cdot x}^{1} \underbrace{+}_{0} a_0$$

  - d multiplications and d additions

Polynomial Evaluation

- **Setting:** ring $R[x]$, can perform basic operations $(+, \times)$ over R
- **Input:** elements $\alpha, a_0, \ldots, a_d \in R$
- **Output:** $p(\alpha)$, where

$$p(x) = a_0 + a_1 x + \cdots + a_d x^d \in R[x]$$

- Naive algorithm:
  - Compute $\alpha^2, \alpha^3, \ldots, \alpha^d$ ($d - 1$ multiplications)
  - Compute $a_j \alpha^j$ (d multiplications)
  - Add $a_j \alpha^j$ (d additions)
- Can we do better?
- Horner's algorithm (a.k.a. Horner's rule):
  - Write

$$p(x) = (\cdots((a_d x + a_{d-1}) \cdot x + a_{d-2}) \cdot x + a_{d-3}) \cdot x + \cdots a_1) \cdot x + a_0$$

  - n multiplications and n additions
- Ostrowski'1954: Is Horner's rule optimal for polynomial evaluation?

Different Cost Function

- From previous lecture, multiplying integers may be harder than adding integers (same problem for matrix rings)
- *Open problem:* is integer addition easier than integer multiplication?

  See resources and final project page of the course to find exciting new developments on this question! In [Harvey, van der Hoeven 2019] the authors find an $O(n \log n)$ algorithm for $\underbrace{\text{multiplying two integers}}_{n\text{-bit integers}}$!

OPEN: is this optimal for multiplication?

OPEN: what is the complexity of multiplying two matrices $(n \times n)$ $\quad (O(n^{\omega}))$ $\quad$ is $\omega = 2$?

add $O(n^2)$

Different Cost Function

- From previous lecture, multiplying integers may be harder than adding integers (same problem for matrix rings)
- *Open problem:* is integer addition easier than integer multiplication?

  See resources and final project page of the course to find exciting new developments on this question! In [Harvey, van der Hoeven 2019] the authors find an $O(n \log n)$ algorithm for multiplying two integers!

- Ostrowski's *non-scalar complexity*
  1. $\mathbb{F}$ is a field, $R = \mathbb{F}[\alpha, a_0, \ldots, a_d]$

Different Cost Function

- From previous lecture, multiplying integers may be harder than adding integers (same problem for matrix rings)
- *Open problem:* is integer addition easier than integer multiplication?

  See resources and final project page of the course to find exciting new developments on this question! In [Harvey, van der Hoeven 2019] the authors find an $O(n \log n)$ algorithm for multiplying two integers!

- Ostrowski's *non-scalar complexity*
  1. $\mathbb{F}$ is a field, $R = \mathbb{F}[\alpha, a_0, \ldots, a_d]$
  2. **Scalar operations:** addition of two elements from R, multiplication of element from R by fixed constant from $\mathbb{F}$ (fixed by algorithm).
  3. **Non-scalar operations:** all other operations

Different Cost Function

- From previous lecture, multiplying integers may be harder than adding integers (same problem for matrix rings)
- *Open problem:* is integer addition easier than integer multiplication?

  See resources and final project page of the course to find exciting new developments on this question! In [Harvey, van der Hoeven 2019] the authors find an $O(n \log n)$ algorithm for multiplying two integers!

- Ostrowski's *non-scalar complexity*
  1. $\mathbb{F}$ is a field, $R = \mathbb{F}[\alpha, a_0, \ldots, a_d]$
  2. **Scalar operations:** addition of two elements from R, multiplication of element from R by fixed constant from $\mathbb{F}$ (fixed by algorithm).
  3. **Non-scalar operations:** all other operations
- Can one improve the non-scalar operations in polynomial evaluation?

Different Cost Function

- From previous lecture, multiplying integers may be harder than adding integers (same problem for matrix rings)
- *Open problem:* is integer addition easier than integer multiplication?

  See resources and final project page of the course to find exciting new developments on this question! In [Harvey, van der Hoeven 2019] the authors find an $O(n \log n)$ algorithm for multiplying two integers!
- Ostrowski's *non-scalar complexity*
  1. $\mathbb{F}$ is a field, $R = \mathbb{F}[\alpha, a_0, \ldots, a_d]$
  2. **Scalar operations:** addition of two elements from R, multiplication of element from R by fixed constant from $\mathbb{F}$ (fixed by algorithm).
  3. **Non-scalar operations:** all other operations
- Can one improve the non-scalar operations in polynomial evaluation?
- [Pan 1966] No! Horner's rule is optimal! } lower bound

input: α , $a_0, \ldots, a_d$

Evaluating a fixed polynomial

- Not all polynomials are created equal.
- What if we want to evaluate a particular polynomial? Say we know coefficients $a_0, a_1, \ldots, a_d \in \mathbb{F}$ and

$$p(x) = a_0 + a_1 x + \cdots a_d x^d \in \mathbb{F}[x]$$

$a_0, \ldots, a_d$ are scalars

Evaluating a fixed polynomial

- Not all polynomials are created equal.
- What if we want to evaluate a particular polynomial? Say we know coefficients $a_0, a_1, \ldots, a_d \in \mathbb{F}$ and

$$p(x) = a_0 + a_1 x + \cdots a_d x^d \in \mathbb{F}[x]$$

- **Input:** value $\alpha \in \mathbb{F}$
- **Output:** evaluation $p(\alpha)$

Evaluating a fixed polynomial

- Not all polynomials are created equal.
- What if we want to evaluate a particular polynomial? Say we know coefficients $a_0, a_1, \ldots, a_d \in \mathbb{F}$ and

$$p(x) = a_0 + a_1 x + \cdots a_d x^d \in \mathbb{F}[x]$$

- **Input:** value $\alpha \in \mathbb{F}$
- **Output:** evaluation $p(\alpha)$

Horner d multiplications

- [Paterson, Stockmeyer 1973]: $p(\alpha)$ can be evaluated with $2\lceil\sqrt{d}\rceil - 1$ non-scalar multiplications.
  - partition p into $\sqrt{d}$ blocks of length $\sqrt{d}$. Say $m = \lceil\sqrt{d}\rceil$ and $k = \lfloor d/m \rfloor + 1$.

$$p(x) = (a_{km-1}x^{m-1} + \cdots + a_{(k-1)m}) \cdot x^{(k-1)m} + \cdots$$
$$\cdots + \underbrace{(a_{2m-1}x^{m-1} + \cdots + a_m)}_{\sqrt{d}\text{ monomials}} \cdot x^m + \underbrace{(a_{m-1}x^{m-1} + \cdots + a_0)}_{\sqrt{d}\text{ monomials}}$$

$km-1 > d \quad a_{km-1} = 0$

Evaluating a fixed polynomial

- Not all polynomials are created equal.
- What if we want to evaluate a particular polynomial? Say we know coefficients $a_0, a_1, \dots, a_d \in \mathbb{F}$ and

$$p(x) = a_0 + a_1 x + \cdots a_d x^d \in \mathbb{F}[x]$$

- **Input:** value $\alpha \in \mathbb{F}$
- **Output:** evaluation $p(\alpha)$
- [Paterson, Stockmeyer 1973]: $p(\alpha)$ can be evaluated with $2\lceil\sqrt{d}\rceil - 1$ non-scalar multiplications.
  - partition p into $\sqrt{d}$ blocks of length $\sqrt{d}$. Say $m = \lceil\sqrt{n}\rceil$ and $k = \lfloor d/m \rfloor + 1$.

$$\begin{aligned} p(x) = {} & (a_{km-1}x^{m-1} + \cdots + a_{(k-1)m}) \cdot x^{(k-1)m} + \cdots \\ & \cdots + (a_{2m-1}x^{m-1} + \cdots + a_m) \cdot x^m + (a_{m-1}x^{m-1} + \cdots + a_0) \end{aligned}$$

  - Compute $\alpha, \underbrace{\alpha^2, \dots, \alpha^m}$ using $m-1$ non-scalar operations

Evaluating a fixed polynomial

- Not all polynomials are created equal.
- What if we want to evaluate a particular polynomial? Say we know coefficients $a_0, a_1, \ldots, a_d \in \mathbb{F}$ and

$$p(x) = a_0 + a_1 x + \cdots a_d x^d \in \mathbb{F}[x]$$

- **Input:** value $\alpha \in \mathbb{F}$
- **Output:** evaluation $p(\alpha)$
- [Paterson, Stockmeyer 1973]: $p(\alpha)$ can be evaluated with $2\lceil\sqrt{d}\rceil - 1$ non-scalar multiplications.
  - partition p into $\sqrt{d}$ blocks of length $\sqrt{d}$. Say $m = \lceil\sqrt{n}\rceil$ and $k = \lfloor d/m \rfloor + 1$.

$$p(x) = \overbrace{\underline{(a_{km-1}x^{m-1} + \cdots + a_{(k-1)m})}}^{\beta_k} \cdot x^{(k-1)m} + \cdots$$
$$\cdots + \underbrace{(a_{2m-1}x^{m-1} + \cdots + a_m)}_{\beta_2} \cdot x^m + \underbrace{(a_{m-1}x^{m-1} + \cdots + a_0)}_{\beta_1}$$

  - Compute $\alpha, \alpha^2, \ldots, \alpha^m$ using $m-1$ non-scalar operations

baby step

  - Compute $\beta_j = \underbrace{a_{jm-1}}_{\text{scalar}}\alpha^{m-1} + \cdots + a_{(j-1)m}$ no cost (scalar ops)

Evaluating a fixed polynomial

- Not all polynomials are created equal.
- What if we want to evaluate a particular polynomial? Say we know coefficients $a_0, a_1, \ldots, a_d \in \mathbb{F}$ and

$$p(x) = a_0 + a_1 x + \cdots a_d x^d \in \mathbb{F}[x]$$

- **Input:** value $\alpha \in \mathbb{F}$
- **Output:** evaluation $p(\alpha)$
- [Paterson, Stockmeyer 1973]: $p(\alpha)$ can be evaluated with $2\lceil\sqrt{d}\rceil - 1$ non-scalar multiplications.
  - partition p into $\sqrt{d}$ blocks of length $\sqrt{d}$. Say $m = \lceil\sqrt{d}\rceil$ and $k = \lfloor d/m \rfloor + 1$.

$$\begin{aligned} p(x) = {} & (a_{km-1}x^{m-1} + \cdots + a_{(k-1)m}) \cdot x^{(k-1)m} + \cdots \\ & \cdots + (a_{2m-1}x^{m-1} + \cdots + a_m) \cdot x^m + (a_{m-1}x^{m-1} + \cdots + a_0) \end{aligned}$$

  - Compute $\alpha, \alpha^2, \ldots, \alpha^m$ using $m-1$ non-scalar operations
  - Compute $\beta_j = a_{km-1}\alpha^{m-1} + \cdots + a_{(k-1)m}$ no cost (scalar ops)
  - Horner's rule on $\sum_{j=0}^{k} \beta_j \cdot \alpha^{jm}$ $k-1$ non-scalar

giant step

$(\alpha^m)^j$ $(\alpha^m)^k$

Evaluating a fixed polynomial

- Not all polynomials are created equal.
- What if we want to evaluate a particular polynomial? Say we know coefficients $a_0, a_1, \ldots, a_d \in \mathbb{F}$ and

$$p(x) = a_0 + a_1 x + \cdots a_d x^d \in \mathbb{F}[x]$$

- **Input:** value $\alpha \in \mathbb{F}$
- **Output:** evaluation $p(\alpha)$
- [Paterson, Stockmeyer 1973]: $p(\alpha)$ can be evaluated with $2\lceil\sqrt{d}\rceil - 1$ non-scalar multiplications.
  - partition p into $\sqrt{d}$ blocks of length $\sqrt{d}$. Say $m = \lceil\sqrt{n}\rceil$ and $k = \lfloor d/m \rfloor + 1$.

$$p(x) = (a_{km-1}x^{m-1} + \cdots + a_{(k-1)m}) \cdot x^{(k-1)m} + \cdots$$
$$\cdots + (a_{2m-1}x^{m-1} + \cdots + a_m) \cdot x^m + (a_{m-1}x^{m-1} + \cdots + a_0)$$

$\approx 2\sqrt{d}$

  - Compute $\alpha, \alpha^2, \ldots, \alpha^m$ using $\boxed{m-1}$ non-scalar operations
  - Compute $\beta_j = a_{km-1}\alpha^{m-1} + \cdots + a_{(k-1)m}$ no cost (scalar ops)
  - Horner's rule on $\sum_{j=0}^{k} \beta_j \cdot \alpha^{jm}$ $\boxed{k-1}$ non-scalar
  - Baby-steps, giant-steps evaluation.

Polynomial Multiplication

- In lecture 1 we saw how to multiply two polynomials of degree d in time $O(d^2)$ (computational model # ring operations)
- Can we do better?

Polynomial Multiplication

- In lecture 1 we saw how to multiply two polynomials of degree d in time $O(d^2)$ (computational model $\#$ ring operations)
- Can we do better?
- YES. Assume $d = 2^k$, and $P, Q \in R[x]$ are of degree $< d$. Let $m = d/2$.

Polynomial Multiplication

- In lecture 1 we saw how to multiply two polynomials of degree d in time $O(d^2)$ (computational model # ring operations)
- Can we do better?
- YES. Assume $d = 2^k$, and $P, Q \in R[x]$ are of degree $< d$. Let $m = d/2$.
- Rewrite:

$$P(x) = \underbrace{P_1(x) \cdot x^m}_{\text{high degree}} + \underbrace{P_0}_{\text{low degree}} \qquad Q(x) = Q_1(x) \cdot x^m + Q_0(x)$$

P_0, P_1, Q_0, Q_1 polynomials of degree $< m = \frac{d}{2}$

Polynomial Multiplication

- In lecture 1 we saw how to multiply two polynomials of degree d in time $O(d^2)$ (computational model # ring operations)
- Can we do better?
- YES. Assume $d = 2^k$, and $P, Q \in R[x]$ are of degree $< d$. Let $m = d/2$.
- Rewrite:

$$P(x) = P_1(x) \cdot x^m + P_0 \qquad Q(x) = Q_1(x) \cdot x^m + Q_0(x)$$

- Now

$$P(x) \cdot Q(x) = \overbrace{P_1 Q_1}^{\text{high}} x^d + \overbrace{(P_1 Q_0 + P_0 Q_1)}^{\text{medium}} x^m + \overbrace{P_0 Q_0}^{\text{low degree}}$$

Polynomial Multiplication

- In lecture 1 we saw how to multiply two polynomials of degree d in time $O(d^2)$ (computational model # ring operations)
- Can we do better?
- YES. Assume $d = 2^k$, and $P, Q \in R[x]$ are of degree $< d$. Let $m = d/2$.
- Rewrite:

$$P(x) = P_1(x) \cdot x^m + P_0 \qquad Q(x) = Q_1(x) \cdot x^m + Q_0(x)$$

- Now

$$P(x) \cdot Q(x) = P_1 Q_1 x^d + (P_1 Q_0 + P_0 Q_1) x^m + P_0 Q_0$$

- Reduce multiplication of two polynomials of degree $< d$ to 4 multiplications of polynomials of degree $< d/2$

$T(d) \leq 4 \cdot T(d/2) + O(d) \Rightarrow O(d^2)$

Polynomial Multiplication

- In lecture 1 we saw how to multiply two polynomials of degree d in time $O(d^2)$ (computational model # ring operations)
- Can we do better?
- YES. Assume $d = 2^k$, and $P, Q \in R[x]$ are of degree $< d$. Let $m = d/2$.
- Rewrite:

$$P(x) = P_1(x) \cdot x^m + P_0 \qquad Q(x) = Q_1(x) \cdot x^m + Q_0(x)$$

- Now

$$P(x) \cdot Q(x) = P_1 Q_1 x^d + (P_1 Q_0 + P_0 Q_1) x^m + P_0 Q_0$$

- Reduce multiplication of two polynomials of degree $< d$ to 4 multiplications of polynomials of degree $< d/2$
- Following master's theorem, this does not help us...

Karatsuba & Ofman's trick (1965)

- Can we reduce number of multiplications (perhaps at the cost of doing more additions)?

Karatsuba & Ofman's trick (1965)

$d = 2m$

- Can we reduce number of multiplications (perhaps at the cost of doing more additions)?
- YES!

$$PQ = P_1 Q_1 (x^d - x^m) + (P_1 + P_0)(Q_1 + Q_0) x^m + P_0 Q_0 (1 - x^m)$$

$$PQ = P_1 Q_1 x^d + \underbrace{(P_1 Q_0 + P_0 Q_1)}_{(P_1 + P_0)(Q_1 + Q_0)} x^m + P_0 Q_0$$

extra term: $P_1 Q_1 x^m + P_0 Q_0 \cdot x^m$

Karatsuba & Ofman's trick (1965)

- Can we reduce number of multiplications (perhaps at the cost of doing more additions)?
- YES!

$$PQ = P_1 Q_1 (x^d - x^m) + (P_1 + P_0)(Q_1 + Q_0) x^m + P_0 Q_0 (1 - x^m)$$

- **Remark:** multiplication by power of x doesn't count as multiplication, as this only shifts the coefficients of the polynomial.

Karatsuba & Ofman's trick (1965)

- Can we reduce number of multiplications (perhaps at the cost of doing more additions)?
- YES!

$$PQ = P_1 Q_1(x^d - x^m) + (P_1 + P_0)(Q_1 + Q_0)x^m + P_0 Q_0(1 - x^m)$$

- **Remark:** multiplication by power of x doesn't count as multiplication, as this only shifts the coefficients of the polynomial.
- Now we have reduced multiplication of two polynomials of degree $< d$ to *3* multiplications of polynomials of degree $< d/2$

Karatsuba & Ofman's trick (1965)

- Can we reduce number of multiplications (perhaps at the cost of doing more additions)?
- YES!

$$PQ = P_1 Q_1 (x^d - x^m) + (P_1 + P_0)(Q_1 + Q_0)x^m + P_0 Q_0 (1 - x^m)$$

- **Remark:** multiplication by power of x doesn't count as multiplication, as this only shifts the coefficients of the polynomial.
- Now we have reduced multiplication of two polynomials of degree $< d$ to *3* multiplications of polynomials of degree $< d/2$
- By master's theorem, we get that Karatsuba-Ofman method can be done with $O(d^{\log_2 3}) = O(d^{1.59})$ ring operations.

previously $O(d^2)$

Complexity of Karatsuba-Ofman

- If $T(2^k) \leq 3 \cdot T(2^{k-1}) + c \cdot 2^k$ then $T(2^k) \leq 3^k - 2c \cdot 2^k$ for $k \geq 1$.

$T(d)$ time to multiply 2 poly deg $< d$

Proof is by induction.

Complexity of Karatsuba-Ofman

- If $T(2^k) \leq T(2^{k-1}) + c \cdot 2^k$ then $\boxed{T(2^k) \leq 3^k - 2c \cdot 2^k}$ for $k \geq 1$.
- $$3^{\log_2 d} = 2^{\log_2 3 \cdot \log_2 d} = d^{\log_2 3}$$

$$T(2^k) \leq 3^k - 2c \cdot 2^k$$

$k = \log_2 d \qquad 2^k = d$

$$T(d) \leq 3^{\log_2 d} = 2^{\log 3 \cdot \log d} = d^{\log_2 3}$$

- Polynomial Evaluation
- Polynomial Multiplication
- Polynomial Interpolation
- Conclusion
- Acknowledgements

Polynomial Interpolation

field

- **Problem:** given $d+1$ evaluations of a polynomial $p(x) \in \mathbb{F}[x]$ of degree $\leq d$, can we "reconstruct" the polynomial p (as list of coefficients)?
- **Input:** evaluations $p(u_0), \ldots, p(u_d)$ of polynomial $p(x)$ of degree $\leq d$
- **Output:** coefficients $(p_0, p_1, \ldots, p_d)$ of $p(x)$

input : $(u_0, p(u_0)), (u_1, p(u_1)), \ldots, (u_d, p(u_d))$

$$p(x) = p_0 + p_1 x + \cdots + p_d x^d$$

Polynomial Interpolation

- **Problem:** given $d+1$ evaluations of a polynomial $p(x) \in \mathbb{F}[x]$ of degree $\leq d$, can we "reconstruct" the polynomial p (as list of coefficients)?
- **Input:** evaluations $p(u_0), \ldots, p(u_d)$ of polynomial $p(x)$ of degree $\leq d$
- **Output:** coefficients $(p_0, p_1, \ldots, p_d)$ of $p(x)$
- Closely related to matrix-vector multiplication:

$$\begin{pmatrix} u_0^0 & u_0^1 & \cdots & u_0^d \\ u_1^0 & u_1^1 & \cdots & u_1^d \\ \vdots & \vdots & \vdots & \vdots \\ u_d^0 & u_d^1 & \cdots & u_d^d \end{pmatrix} \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_d \end{pmatrix} = \begin{pmatrix} p(u_0) \\ p(u_1) \\ \vdots \\ p(u_d) \end{pmatrix}$$

$$p(x) = p_0 \cdot x^0 + p_1 x^1 + p_2 x^2 + \cdots + p_d x^d =$$

$$= \underline{(x^0, x^1, \ldots, x^d)} \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_d \end{pmatrix}$$

Polynomial Interpolation

- **Problem:** given $d+1$ evaluations of a polynomial $p(x) \in \mathbb{F}[x]$ of degree $\leq d$, can we "reconstruct" the polynomial p (as list of coefficients)?
- **Input:** evaluations $p(u_0), \ldots, p(u_d)$ of polynomial $p(x)$ of degree $\leq d$
- **Output:** coefficients $(p_0, p_1, \ldots, p_d)$ of $p(x)$
- Closely related to matrix-vector multiplication:

Vandermonde matrix

unknown

$$\boxed{\begin{pmatrix} u_0^0 & u_0^1 & \cdots & u_0^d \\ u_1^0 & u_1^1 & \cdots & u_1^d \\ \vdots & \vdots & \vdots & \vdots \\ u_d^0 & u_d^1 & \cdots & u_d^d \end{pmatrix}} \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_d \end{pmatrix} = \begin{pmatrix} p(u_0) \\ p(u_1) \\ \vdots \\ p(u_d) \end{pmatrix}$$

inputs

- Interpolation amounts to inverting Vandermonde matrix!

$$V \begin{pmatrix} p_0 \\ \vdots \\ p_d \end{pmatrix} = \begin{pmatrix} p(u_0) \\ \vdots \\ p(u_d) \end{pmatrix} \Rightarrow \begin{pmatrix} p_0 \\ \vdots \\ p_d \end{pmatrix} = V^{-1} \begin{pmatrix} p(u_0) \\ \vdots \\ p(u_d) \end{pmatrix}$$

Polynomial Interpolation

distinct

- **Problem:** given $d+1$ evaluations of a polynomial $p(x) \in \boxed{\mathbb{F}[x]}$ of degree $\leq d$, can we "reconstruct" the polynomial p (as list of coefficients)?
- **Input:** evaluations $p(u_0), \ldots, p(u_d)$ of polynomial $p(x)$ of degree $\leq d$
- **Output:** coefficients $(p_0, p_1, \ldots, p_d)$ of $p(x)$
- Closely related to matrix-vector multiplication:

prove that

$\det(V) = \prod_{i>j} (u_i - u_j)$

$\neq 0$ because u_i's are distinct.

$$\begin{pmatrix} u_0^0 & u_0^1 & \cdots & u_0^d \\ u_1^0 & u_1^1 & \cdots & u_1^d \\ \vdots & \vdots & \vdots & \vdots \\ u_d^0 & u_d^1 & \cdots & u_d^d \end{pmatrix} \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_d \end{pmatrix} = \begin{pmatrix} p(u_0) \\ p(u_1) \\ \vdots \\ p(u_d) \end{pmatrix}$$

- Interpolation amounts to inverting Vandermonde matrix!
- Will use this idea later in course to obtain faster algorithm for polynomial multiplication

Polynomial Interpolation

Polynomial Multiplication Non-Scalar Setting

Theorem

Let $\mathbb{F}$ be a field with $\geq 2d+1$ elements. Polynomial multiplication over $\mathbb{F}[x]$ in the non-scalar model of two polynomials of degree $\leq d$ can be done with $2d+1$ non-scalar multiplications.

Karatsuba-Ofman $O(d^{1.59})$ operations

(ring operations)

Polynomial Multiplication Non-Scalar Setting

Theorem

Let $\mathbb{F}$ be a field with $\geq 2d+1$ elements. Polynomial multiplication over $\mathbb{F}[x]$ in the non-scalar model of two polynomials of degree $\leq d$ can be done with $2d+1$ non-scalar multiplications.

- Pick $2d+1$ distinct scalars $u_0, \ldots, u_{2d} \in \mathbb{F}$

(hardcode these values in our machine, optimize operations with them)

Polynomial Multiplication Non-Scalar Setting

Theorem

Let $\mathbb{F}$ be a field with $\geq 2d+1$ elements. Polynomial multiplication over $\mathbb{F}[x]$ in the non-scalar model of two polynomials of degree $\leq d$ can be done with $2d+1$ non-scalar multiplications.

- Pick $2d+1$ distinct scalars $u_0, \ldots, u_{2d} \in \mathbb{F}$
- Evaluate $p(u_i), q(u_i)$. (no cost - only scalar multiplications)

$p(x) = p_0 + p_1 x + \cdots + p_d x^d$

p_i scalars $(\in \mathbb{F})$

u_i scalars

u_i^k scalar

$p_k u_i^\ell$ scalar

Polynomial Multiplication Non-Scalar Setting

Theorem

Let $\mathbb{F}$ be a field with $\geq 2d+1$ elements. Polynomial multiplication over $\mathbb{F}[x]$ in the non-scalar model of two polynomials of degree $\leq d$ can be done with $2d+1$ non-scalar multiplications.

- Pick $2d+1$ distinct scalars $u_0, \ldots, u_{2d} \in \mathbb{F}$
- Evaluate $p(u_i), q(u_i)$. (no cost - only scalar multiplications)
- Compute $\gamma_i = p(u_i)q(u_i)$ ($2d+1$ non-scalar multiplications)

$$\underbrace{p(u_i) \cdot q(u_i)}_{\text{Ostrowski's model: non-scalar operations}} = (p \cdot q)(u_i)$$

$2d+1$ evaluations of $\underbrace{(p \cdot q)(x)}_{\text{degree} \leq 2d}$

interpolation!

Polynomial Multiplication Non-Scalar Setting

Theorem

Let $\mathbb{F}$ be a field with $\geq 2d+1$ elements. Polynomial multiplication over $\mathbb{F}[x]$ in the non-scalar model of two polynomials of degree $\leq d$ can be done with $2d+1$ non-scalar multiplications.

- Pick $2d+1$ distinct scalars $u_0, \ldots, u_{2d} \in \mathbb{F}$
- Evaluate $p(u_i), q(u_i)$. (no cost - only scalar multiplications)
- Compute $\gamma_i = p(u_i)q(u_i)$ ($2d+1$ non-scalar multiplications)
- Lagrange polynomial: (only scalar multiplications)

$$L_i(x) = \prod_{j \neq i} \frac{x - u_j}{u_i - u_j}$$

constants

$$L_i(u_j) = \begin{cases} j \neq i \Rightarrow 0 \\ j = i \Rightarrow L_i(u_i) = 1 \end{cases}$$

Polynomial Multiplication Non-Scalar Setting

Theorem

Let $\mathbb{F}$ be a field with $\geq 2d+1$ elements. Polynomial multiplication over $\mathbb{F}[x]$ in the non-scalar model of two polynomials of degree $\leq d$ can be done with $2d+1$ non-scalar multiplications.

- Pick $2d+1$ distinct scalars $u_0, \ldots, u_{2d} \in \mathbb{F}$
- Evaluate $p(u_i), q(u_i)$. (no cost - only scalar multiplications)
- Compute $\gamma_i = p(u_i)q(u_i)$ (2d + 1 non-scalar multiplications)
- Lagrange polynomial: (only scalar multiplications)

$$\boxed{L_i(x) = \prod_{j \neq i} \frac{x - u_j}{u_i - u_j}}$$

only using scalar multiplications

- $$p(x)q(x) = \sum_{i=0}^{2d} \gamma_i \cdot L_i(x)$$

only scalar multiplications

Polynomial multiplication

$$p(x)\,q(x) = \sum_{i=0}^{2d} \gamma_i L_i(x) \qquad \gamma_i = p(u_i)\,q(u_i)$$

$$p(u_j)\cdot q(u_j) = \sum_{i=0}^{2d} \gamma_i \underbrace{L_i(u_j)}_{\begin{cases} 1 & \text{if } i=j \\ 0 & \text{otherwise} \end{cases}} = \gamma_j \cdot 1$$

two expression of polynomials $\begin{cases} p(x)\cdot q(x) \\ \sum_{i=0}^{2d} \gamma_i L_i(x) \end{cases}$

of degree $\leq 2d$ which agree on $2d+1$ distinct evaluations $\Rightarrow$ these two polynomials are the same!

- Polynomial Evaluation
- Polynomial Multiplication
- Polynomial Interpolation
- Conclusion
- Acknowledgements

Conclusion

In today's lecture, we learned

- computational models for measuring complexity of multiplication, evaluation and interpolation
  - ring operations
  - non-scalar complexity
- Algorithms for
  - polynomial evaluation
  - polynomial multiplication
  - polynomial interpolation

Acknowledgement

- Based largely on Arne's notes

  https://cs.uwaterloo.ca/~r5olivei/courses/2021-winter-cs487/lec3-ref.pdf

References I

Pan, Victor 1966.
Methods for computing values of polynomials
Russian Mathematical Surveys

Paterson, M. and Stockmeyer, L. 1973.
On the number of nonscalar multiplications necessary to evaluate polynomials
SIAM Journal of Computing

Harvey, David and van der Hoeven, Joris 2019.
Integer multiplication in time $O(n \log n)$
Annals of mathematics
https://hal.archives-ouvertes.fr/hal-02070778/document