# CMPSCI 611: Advanced Algorithms

## Lecture 3: Fast Polynomial Multiplication

Andrew McGregor

Last Compiled: February 8, 2024

# Slide Left Blank

# Polynomial Multiplication

Problem: Suppose $A(x)$ and $B(x)$ are polynomials of degree $n-1$:

$$A(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{n-1}x^{n-1}$$
$$B(x) = b_0 + b_1x + b_2x^2 + \ldots + b_{n-1}x^{n-1}$$

Compute $C(x) = A(x)B(x)$. We'll assume $n$ is a power of 2.

# Polynomial Multiplication

Problem: Suppose $A(x)$ and $B(x)$ are polynomials of degree $n-1$:

$$\begin{aligned} A(x) =& a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1} \\ B(x) =& b_0 + b_1 x + b_2 x^2 + \ldots + b_{n-1} x^{n-1} \end{aligned}$$

Compute $C(x) = A(x)B(x)$. We'll assume $n$ is a power of 2.

How long does naive algorithm take?

# Polynomial Multiplication

Problem: Suppose $A(x)$ and $B(x)$ are polynomials of degree $n-1$:

$$A(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$$
$$B(x) = b_0 + b_1 x + b_2 x^2 + \ldots + b_{n-1} x^{n-1}$$

Compute $C(x) = A(x)B(x)$. We'll assume $n$ is a power of 2.

How long does naive algorithm take? $O(n^2)$

# Representation of Polynomials

## Definition

The *coefficient representation* (CR) of a polynomial the vector of coefficients. E.g., $(1, 3, -2, 1)$ is the coefficient representation of

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

# Representation of Polynomials

## Definition

The *coefficient representation* (CR) of a polynomial the vector of coefficients. E.g., $(1, 3, -2, 1)$ is the coefficient representation of

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

## Definition

The *point-value representation* (PVR) of a polynomial: for $n$ distinct points $x_0, \ldots, x_{n-1}$ the PVR of $f$ is

$$\{(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_{n-1}, f(x_{n-1}))\}$$

E.g., $f(x) \equiv \{(0, 1), (1, 3), (2, 7), (3, 19)\}$.

# Representation of Polynomials

## Definition

The *coefficient representation* (CR) of a polynomial the vector of coefficients. E.g., $(1, 3, -2, 1)$ is the coefficient representation of

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

## Definition

The *point-value representation* (PVR) of a polynomial: for $n$ distinct points $x_0, \ldots, x_{n-1}$ the PVR of $f$ is

$$\{(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_{n-1}, f(x_{n-1}))\}$$

E.g., $f(x) \equiv \{(0,1), (1,3), (2,7), (3,19)\}$.

## Lemma

*Specifying the value of a function at $n$ distinct points uniquely specifies a degree $n-1$ polynomial that goes through those points.*

# Polynomial Arithmetic in Point-Value Representation

- First attempt: Let $x_0, \ldots, x_{n-1}$ be distinct and suppose

$$A(x) \equiv \{(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\}$$
$$B(x) \equiv \{(x_0, z_0), (x_1, z_1), \ldots, (x_{n-1}, z_{n-1})\}$$

# Polynomial Arithmetic in Point-Value Representation

- First attempt: Let $x_0, \ldots, x_{n-1}$ be distinct and suppose

$$A(x) \equiv \{(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\}$$
$$B(x) \equiv \{(x_0, z_0), (x_1, z_1), \ldots, (x_{n-1}, z_{n-1})\}$$

- Then surely,

$$C(x) \equiv \{(x_0, y_0 z_0), (x_1, y_1 z_1), \ldots, (x_{n-1}, y_{n-1} z_{n-1})\}$$

# Polynomial Arithmetic in Point-Value Representation

- First attempt: Let $x_0, \ldots, x_{n-1}$ be distinct and suppose

$$A(x) \equiv \{(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\}$$
$$B(x) \equiv \{(x_0, z_0), (x_1, z_1), \ldots, (x_{n-1}, z_{n-1})\}$$

- Then surely,

$$C(x) \equiv \{(x_0, y_0 z_0), (x_1, y_1 z_1), \ldots, (x_{n-1}, y_{n-1} z_{n-1})\}$$

- Issue: While $C(x_i) = y_i z_i$, $C$ is a degree $2n - 2$ polynomial and we need $2n - 1$ distinct points to specify it.

# Polynomial Arithmetic in Point-Value Representation

- First attempt: Let $x_0, \ldots, x_{n-1}$ be distinct and suppose

$$A(x) \equiv \{(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\}$$
$$B(x) \equiv \{(x_0, z_0), (x_1, z_1), \ldots, (x_{n-1}, z_{n-1})\}$$

- Then surely,

$$C(x) \equiv \{(x_0, y_0 z_0), (x_1, y_1 z_1), \ldots, (x_{n-1}, y_{n-1} z_{n-1})\}$$

- Issue: While $C(x_i) = y_i z_i$, $C$ is a degree $2n-2$ polynomial and we need $2n-1$ distinct points to specify it.
- Fix: Assume $A$ and $B$ are specified on at least $2n-1$ distinct points.

# Polynomial Arithmetic in Point-Value Representation

- First attempt: Let $x_0, \ldots, x_{n-1}$ be distinct and suppose

$$A(x) \equiv \{(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\}$$
$$B(x) \equiv \{(x_0, z_0), (x_1, z_1), \ldots, (x_{n-1}, z_{n-1})\}$$

- Then surely,

$$C(x) \equiv \{(x_0, y_0 z_0), (x_1, y_1 z_1), \ldots, (x_{n-1}, y_{n-1} z_{n-1})\}$$

- Issue: While $C(x_i) = y_i z_i$, $C$ is a degree $2n-2$ polynomial and we need $2n-1$ distinct points to specify it.
- Fix: Assume $A$ and $B$ are specified on at least $2n-1$ distinct points.
- Can compute PVR of $C$ is $\Theta(n)$ time. But what about coefficient representation?

# Framework for Fast Polynomial Multiplication

- Input: Coefficient representation of $A(x)$ and $B(x)$

# Framework for Fast Polynomial Multiplication

- Input: Coefficient representation of $A(x)$ and $B(x)$
- Step 1: Transform into PVR by evaluating on at least $2n-1$ points

# Framework for Fast Polynomial Multiplication

- Input: Coefficient representation of $A(x)$ and $B(x)$
- Step 1: Transform into PVR by evaluating on at least $2n - 1$ points
- Step 2: Multiply polynomials to get $C(x)$ in PVR

# Framework for Fast Polynomial Multiplication

- Input: Coefficient representation of $A(x)$ and $B(x)$
- Step 1: Transform into PVR by evaluating on at least $2n - 1$ points
- Step 2: Multiply polynomials to get $C(x)$ in PVR
- Step 3: Transform PVR of $C(x)$ back into CR.

# Framework for Fast Polynomial Multiplication

- Input: Coefficient representation of $A(x)$ and $B(x)$
- Step 1: Transform into PVR by evaluating on at least $2n-1$ points
- Step 2: Multiply polynomials to get $C(x)$ in PVR
- Step 3: Transform PVR of $C(x)$ back into CR.

Naive implementation of step 1 takes

# Framework for Fast Polynomial Multiplication

- Input: Coefficient representation of $A(x)$ and $B(x)$
- Step 1: Transform into PVR by evaluating on at least $2n-1$ points
- Step 2: Multiply polynomials to get $C(x)$ in PVR
- Step 3: Transform PVR of $C(x)$ back into CR.

Naive implementation of step 1 takes $O(n^2)$ time.

# Framework for Fast Polynomial Multiplication

- Input: Coefficient representation of $A(x)$ and $B(x)$
- Step 1: Transform into PVR by evaluating on at least $2n - 1$ points
- Step 2: Multiply polynomials to get $C(x)$ in PVR
- Step 3: Transform PVR of $C(x)$ back into CR.

Naive implementation of step 1 takes $O(n^2)$ time. We'll do steps 1 and 3 in $O(n \log n)$ time.

# Framework for Fast Polynomial Multiplication

- Input: Coefficient representation of $A(x)$ and $B(x)$
- Step 1: Transform into PVR by evaluating on at least $2n-1$ points
- Step 2: Multiply polynomials to get $C(x)$ in PVR
- Step 3: Transform PVR of $C(x)$ back into CR.

Naive implementation of step 1 takes $O(n^2)$ time. We'll do steps 1 and 3 in $O(n \log n)$ time.

Important: We can choose any distinct points for the PVR. Let's use the complex roots of unity...

# Complex Roots of Unity

## Definition

The $n$-th roots of unity are the complex solutions to the equation $x^n = 1$, i.e.,

$$e^{2\pi ik/n} = \cos\frac{2\pi k}{n} + i\sin\frac{2\pi k}{n} \qquad k = 0, \ldots, n-1.$$

Let $\omega_n = e^{2\pi i/n}$.

# Complex Roots of Unity

## Definition

The $n$-th roots of unity are the complex solutions to the equation $x^n = 1$, i.e.,

$$e^{2\pi ik/n} = \cos\frac{2\pi k}{n} + i\sin\frac{2\pi k}{n} \qquad k = 0, \ldots, n-1.$$

Let $\omega_n = e^{2\pi i/n}$.

## Lemma (Halving Lemma)

*The squares of the* $2n$*-th roots of unity are two copies of the* $n$*-th roots of unity:*

$$\{(\omega_{2n}^0)^2, \ldots, (\omega_{2n}^{2n-1})^2\} = \{\omega_n^0, \ldots, \omega_n^{n-1}\} \cup \{\omega_n^0, \ldots, \omega_n^{n-1}\}$$

# Complex Roots of Unity

## Definition

The $n$-th roots of unity are the complex solutions to the equation $x^n = 1$, i.e.,

$$e^{2\pi ik/n} = \cos\frac{2\pi k}{n} + i\sin\frac{2\pi k}{n} \qquad k = 0, \ldots, n-1.$$

Let $\omega_n = e^{2\pi i/n}$.

## Lemma (Halving Lemma)

*The squares of the $2n$-th roots of unity are two copies of the $n$-th roots of unity:*

$$\{(\omega_{2n}^0)^2, \ldots, (\omega_{2n}^{2n-1})^2\} = \{\omega_n^0, \ldots, \omega_n^{n-1}\} \cup \{\omega_n^0, \ldots, \omega_n^{n-1}\}$$

## Proof.

Follows since $(\omega_{2n}^r)^2 = e^{2r\cdot 2\pi i/(2n)} = e^{r\cdot 2\pi i/n} = \omega_n^r$ and $(\omega_{2n}^{r+n})^2 = \omega_n^r$. □

# Divide and Conquer for Polynomial Evaluation

- Write degree $n-1$ polynomial to be evaluated in terms of two degree $n/2-1$ polynomials:

$$A(x) \quad = \quad a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$$

# Divide and Conquer for Polynomial Evaluation

- Write degree $n-1$ polynomial to be evaluated in terms of two degree $n/2-1$ polynomials:

$$\begin{aligned} A(x) &= a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1} \\ &= (a_0 + a_2 x^2 + \ldots + a_{n-2} x^{n-2}) \\ &\qquad + x(a_1 + a_3 x^2 + \ldots + a_{n-1} x^{n-2}) \end{aligned}$$

# Divide and Conquer for Polynomial Evaluation

- Write degree $n-1$ polynomial to be evaluated in terms of two degree $n/2-1$ polynomials:

$$\begin{aligned} A(x) &= a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1} \\ &= (a_0 + a_2 x^2 + \ldots + a_{n-2} x^{n-2}) \\ &\qquad + x(a_1 + a_3 x^2 + \ldots + a_{n-1} x^{n-2}) \\ &= A_{\text{even}}(x^2) + x A_{\text{odd}}(x^2) \end{aligned}$$

# Divide and Conquer for Polynomial Evaluation

- Write degree $n-1$ polynomial to be evaluated in terms of two degree $n/2-1$ polynomials:

$$\begin{aligned} A(x) &= a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1} \\ &= (a_0 + a_2 x^2 + \ldots + a_{n-2} x^{n-2}) \\ &\qquad + x(a_1 + a_3 x^2 + \ldots + a_{n-1} x^{n-2}) \\ &= A_{\text{even}}(x^2) + x A_{\text{odd}}(x^2) \end{aligned}$$

- To evaluate $A$ at $2n$-th roots of unity, we evaluate $A_{\text{even}}$ and $A_{\text{odd}}$ at $x^2$ for

$$x \in \{\omega_{2n}^0, \omega_{2n}^1, \ldots, \omega_{2n}^{2n-1}\}$$

# Divide and Conquer for Polynomial Evaluation

- Write degree $n-1$ polynomial to be evaluated in terms of two degree $n/2-1$ polynomials:

$$\begin{aligned} A(x) &= a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1}x^{n-1} \\ &= (a_0 + a_2 x^2 + \ldots + a_{n-2}x^{n-2}) \\ &\qquad + x(a_1 + a_3 x^2 + \ldots + a_{n-1}x^{n-2}) \\ &= A_{\text{even}}(x^2) + xA_{\text{odd}}(x^2) \end{aligned}$$

- To evaluate $A$ at $2n$-th roots of unity, we evaluate $A_{\text{even}}$ and $A_{\text{odd}}$ at $x^2$ for

$$x \in \{\omega_{2n}^0, \omega_{2n}^1, \ldots, \omega_{2n}^{2n-1}\}$$

- If $T(n)$ is time to evaluate degree $n-1$ poly at $2n$-th roots of unity,

$$T(1) = \Theta(1) \quad \text{and} \quad T(n) = 2T(n/2) + \Theta(n)$$

# Divide and Conquer for Polynomial Evaluation

- Write degree $n-1$ polynomial to be evaluated in terms of two degree $n/2-1$ polynomials:

$$\begin{aligned}
A(x) &= a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1}x^{n-1} \\
&= (a_0 + a_2 x^2 + \ldots + a_{n-2}x^{n-2}) \\
&\qquad + x(a_1 + a_3 x^2 + \ldots + a_{n-1}x^{n-2}) \\
&= A_{\text{even}}(x^2) + x A_{\text{odd}}(x^2)
\end{aligned}$$

- To evaluate $A$ at $2n$-th roots of unity, we evaluate $A_{\text{even}}$ and $A_{\text{odd}}$ at $x^2$ for

$$x \in \{\omega_{2n}^0, \omega_{2n}^1, \ldots, \omega_{2n}^{2n-1}\}$$

- If $T(n)$ is time to evaluate degree $n-1$ poly at $2n$-th roots of unity,

$$T(1) = \Theta(1) \quad \text{and} \quad T(n) = 2T(n/2) + \Theta(n)$$

- Use Master Theorem to conclude that $T(n) = \Theta(n \log n)$.

# Back to Framework...

- Input: Coefficient representation of $A(x)$ and $B(x)$
- Step 1: Transform into PVR by evaluating at at least $2n - 1$ points
- Step 2: Multiply polynomials to get $C(x)$ in PVR
- Step 3: Transform PVR of $C(x)$ back into CR.

# Back to Framework...

- Input: Coefficient representation of $A(x)$ and $B(x)$
- Step 1: Transform into PVR by evaluating at at least $2n - 1$ points
- Step 2: Multiply polynomials to get $C(x)$ in PVR
- Step 3: Transform PVR of $C(x)$ back into CR.

We now know:

1. Step 1 can be done in $O(n \log n)$ time.
2. Step 2 can be done in $O(n)$ time.

# Back to Framework...

- Input: Coefficient representation of $A(x)$ and $B(x)$
- Step 1: Transform into PVR by evaluating at at least $2n - 1$ points
- Step 2: Multiply polynomials to get $C(x)$ in PVR
- Step 3: Transform PVR of $C(x)$ back into CR.

We now know:

1. Step 1 can be done in $O(n \log n)$ time.
2. Step 2 can be done in $O(n)$ time.

It turns out that Step 3 is almost identical to Step 1!

# Polynomial Evaluation and Interpolation

Step 1 Revisited: Transform $(a_0, a_1, \ldots, a_{n-1})$ to

$$\{(\omega_{2n}^0, y_0), (\omega_{2n}^1, y_1), \ldots, (\omega_{2n}^{2n-1}, y_{2n-1})\}$$

where $y_i = A(\omega_{2n}^i)$.

# Polynomial Evaluation and Interpolation

Step 1 Revisited: Transform $(a_0, a_1, \ldots, a_{n-1})$ to

$$\{(\omega_{2n}^0, y_0), (\omega_{2n}^1, y_1), \ldots, (\omega_{2n}^{2n-1}, y_{2n-1})\}$$

where $y_i = A(\omega_{2n}^i)$. In other words, we need to evaluate:

$$V_n \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{2n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{2n-1} \end{pmatrix}$$

where $a_i = 0$ for $i \geq n-1$ and

$$V_n = \begin{pmatrix} 1 & 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega_{2n} & \omega_{2n}^2 & \omega_{2n}^3 & \ldots & \omega_{2n}^{2n-1} \\ 1 & \omega_{2n}^2 & \omega_{2n}^4 & \omega_{2n}^6 & \ldots & \omega_{2n}^{2(2n-1)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega_{2n}^{2n-1} & \omega_{2n}^{2(2n-1)} & \omega_{2n}^{3(2n-1)} & \ldots & \omega_{2n}^{(2n-1)(2n-1)} \end{pmatrix}$$

# Polynomial Evaluation and Interpolation

Step 3 as inverse of Step 1: Need to transform

$$\{(\omega_{2n}^0, y_0), (\omega_{2n}^1, y_1), \ldots, (\omega_{2n}^{2n-1}, y_{2n-1})\}$$

into $(a_0, a_1, \ldots, a_{2n-1})$ where $y_i = A(\omega_{2n}^i)$.

# Polynomial Evaluation and Interpolation

Step 3 as inverse of Step 1: Need to transform

$$\{(\omega_{2n}^0, y_0), (\omega_{2n}^1, y_1), \ldots, (\omega_{2n}^{2n-1}, y_{2n-1})\}$$

into $(a_0, a_1, \ldots, a_{2n-1})$ where $y_i = A(\omega_{2n}^i)$. In other words, we need

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{2n-1} \end{pmatrix} = V_n^{-1} \cdot \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{2n-1} \end{pmatrix}$$

# Polynomial Evaluation and Interpolation

Step 3 as inverse of Step 1: Need to transform

$$\{(\omega_{2n}^0, y_0), (\omega_{2n}^1, y_1), \ldots, (\omega_{2n}^{2n-1}, y_{2n-1})\}$$

into $(a_0, a_1, \ldots, a_{2n-1})$ where $y_i = A(\omega_{2n}^i)$. In other words, we need

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{2n-1} \end{pmatrix} = V_n^{-1} \cdot \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{2n-1} \end{pmatrix}$$

The inverse of $V_n$ is just $V_n$ with $\omega_{2n}$ replaced by $\omega_{2n}^{-1}$

$$V_n^{-1} = \frac{1}{2n} \begin{pmatrix} 1 & 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega_{2n}^{-1} & \omega_{2n}^{-2} & \omega_{2n}^{-3} & \ldots & \omega_{2n}^{-(2n-1)} \\ 1 & \omega_{2n}^{-2} & \omega_{2n}^{-4} & \omega_{2n}^{-6} & \ldots & \omega_{2n}^{-2(2n-1)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega_{2n}^{-(2n-1)} & \omega_{2n}^{-2(2n-1)} & \omega_{2n}^{-3(2n-1)} & \ldots & \omega_{2n}^{-(2n-1)(2n-1)} \end{pmatrix}$$

# Solving Step 3 Outline

# Solving Step 3 Outline

- Need to compute:

$$a_k = \frac{\hat{A}(\omega_{2n}^{-k})}{2n} \quad \text{for } k = 0, \ldots, 2n-1$$

where $\hat{A}(x) = y_0 + y_1 x + \ldots + y_{2n-1} x^{2n-1}$

# Solving Step 3 Outline

- Need to compute:

$$a_k = \frac{\hat{A}(\omega_{2n}^{-k})}{2n} \quad \text{for } k = 0, \ldots, 2n-1$$

  where $\hat{A}(x) = y_0 + y_1 x + \ldots + y_{2n-1} x^{2n-1}$
- Rewrite $\hat{A}(x) = \hat{A}_{\text{even}}(x^2) + x\hat{A}_{\text{odd}}(x^2)$

# Solving Step 3 Outline

- Need to compute:

$$a_k = \frac{\hat{A}(\omega_{2n}^{-k})}{2n} \quad \text{for } k = 0, \ldots, 2n-1$$

  where $\hat{A}(x) = y_0 + y_1 x + \ldots + y_{2n-1} x^{2n-1}$
- Rewrite $\hat{A}(x) = \hat{A}_{\mathrm{even}}(x^2) + x\hat{A}_{\mathrm{odd}}(x^2)$
- To evaluate $\hat{A}$ on

$$\{\omega_{2n}^{0}, \omega_{2n}^{-1}, \ldots, \omega_{2n}^{-(2n-1)}\}$$

  it suffices to evaluate $\hat{A}_{\mathrm{even}}$ and $\hat{A}_{\mathrm{odd}}$ on

$$\{\omega_{n}^{0}, \omega_{n}^{-1}, \ldots, \omega_{n}^{-(n-1)}\}$$

  because Halving Lemma also applies to $\omega_{2n}^{-1}$.

# Solving Step 3 Outline

- Need to compute:

$$a_k = \frac{\hat{A}(\omega_{2n}^{-k})}{2n} \quad \text{for } k = 0, \ldots, 2n-1$$

  where $\hat{A}(x) = y_0 + y_1 x + \ldots + y_{2n-1}x^{2n-1}$
- Rewrite $\hat{A}(x) = \hat{A}_{\mathrm{even}}(x^2) + x\hat{A}_{\mathrm{odd}}(x^2)$
- To evaluate $\hat{A}$ on

$$\{\omega_{2n}^{0}, \omega_{2n}^{-1}, \ldots, \omega_{2n}^{-(2n-1)}\}$$

  it suffices to evaluate $\hat{A}_{\mathrm{even}}$ and $\hat{A}_{\mathrm{odd}}$ on

$$\{\omega_{n}^{0}, \omega_{n}^{-1}, \ldots, \omega_{n}^{-(n-1)}\}$$

  because Halving Lemma also applies to $\omega_{2n}^{-1}$.
- Step 3 can also be done in $O(n \log n)$ steps.

# Slide Left Blank