10 March 2025

The Fast Fourier Transform

Plan

* Polynomial Multiplication & Evaluation Problem
* Announcements
* Multiplying Polynomials via FFT

Polynomial Multiplication

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

$$q(x) = x^3 - x$$

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

$$= 3x^5 + 2x^4 - x^3$$

$$- 3x^3 - 2x^2 + x$$

$$3x^5 + 2x^4 - 4x^3 - 2x^2 + x$$

- Generalizes Integer Multiplication
- Equivalent to Convolution

$$p(x) = \sum_{i=0}^{n} p_i \cdot x^i \qquad q(x) = \sum_{j=0}^{n} q_j x^j$$

$$p \cdot q(x) = \underbrace{\left(\sum_{i=0}^{n} p_i x^i\right)\left(\sum_{j=0}^{n} q_j x^j\right)}_{\Theta(n^2) \text{ cross terms}}$$

$$\vdots$$

$$= \underbrace{c_{2n} x^{2n} + c_{2n-1} x^{2n-1} + \cdots + c_1 x + c_0}_{\Theta(n) \text{ eventual terms}}$$

Representing Polynomials

* Coefficient Representation

Degree-n polynomial $p(x) = \sum_{i=0}^{n} p_i \cdot x^i$

specified uniquely by $p = p_n, p_{n-1}, \ldots, p_1, p_0$

* Point-Value Representation

Fundamental Fact. Every Degree-n polynomial is uniquely determined by its evaluation at n+1 points.

Representing Polynomials

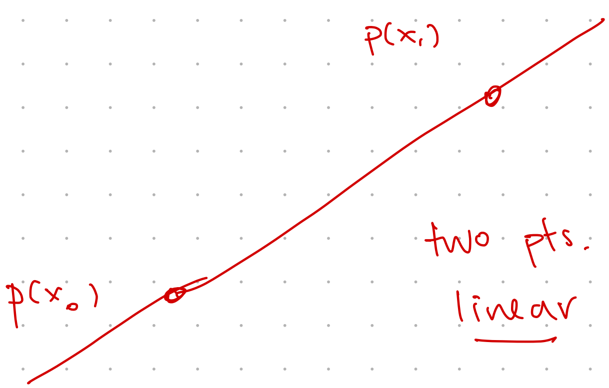
* Coefficient Representation

Degree-n polynomial $p(x) = \sum_{i=0}^{n} p_i \cdot x^i$

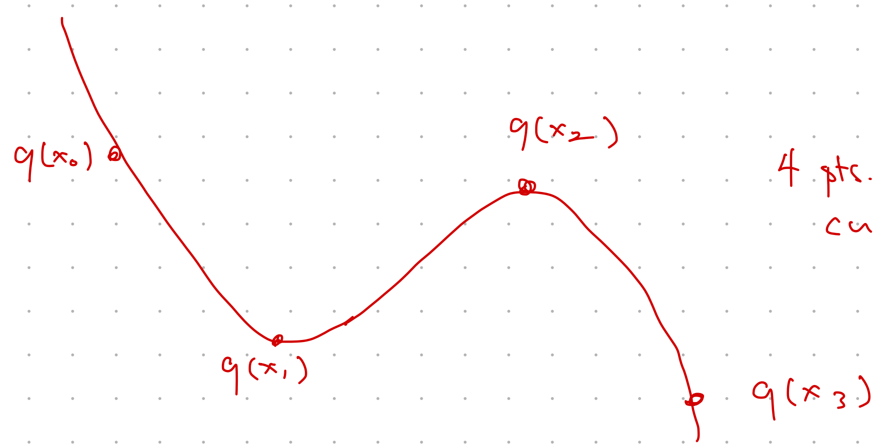
specified uniquely by $p = p_n, p_{n-1}, \ldots, p_1, p_0$

* Point-Value Representation

Fundamental Fact. Every Degree-n polynomial is uniquely determined by its evaluation at $n+1$ points.



two pts. define linear polynomial



4 pts. define cubic

Key Observation: Multiplication is EASY in Point-Value Representation

	$\hat{p}$	$\hat{q}$	$\widehat{p \cdot q}$
x_0	$p(x_0)$	$q(x_0)$	$p(x_0) \times q(x_0)$
x_1	$p(x_1)$	$q(x_1)$	$p(x_1) \times q(x_1)$
x_2	$p(x_2)$	$q(x_2)$	$p(x_2) \times q(x_2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_{2n-1}	$p(x_{2n-1})$	$q(x_{2n-1})$	$p(x_{2n-1}) \times q(x_{2n-1})$
x_{2n}	$p(x_{2n})$	$q(x_{2n})$	$p(x_{2n}) \times q(x_{2n})$

Key Observation: Multiplication is EASY in Point-Value Representation

	$\hat{p}$	$\hat{q}$	$\widehat{p \cdot q}$
x_0	$p(x_0)$	$q(x_0)$	$p(x_0) \times q(x_0)$
x_1	$p(x_1)$	$q(x_1)$	$p(x_1) \times q(x_1)$
x_2	$p(x_2)$	$q(x_2)$	$p(x_2) \times q(x_2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_{2n-1}	$p(x_{2n-1})$	$q(x_{2n-1})$	$p(x_{2n-1}) \times q(x_{2n-1})$
x_{2n}	$p(x_{2n})$	$q(x_{2n})$	$p(x_{2n}) \times q(x_{2n})$

1 multiplication per evaluation point!

$O(n)$ operations

Key Observation: Multiplication is EASY in Point-Value Representation

	$\hat{p}$	$\hat{q}$	$\widehat{p \cdot q}$
x_0	$p(x_0)$	$q(x_0)$	$p(x_0) \times q(x_0)$
x_1	$p(x_1)$	$q(x_1)$	$p(x_1) \times q(x_1)$
x_2	$p(x_2)$	$q(x_2)$	$p(x_2) \times q(x_2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_{2n-1}	$p(x_{2n-1})$	$q(x_{2n-1})$	$p(x_{2n-1}) \times q(x_{2n-1})$
x_{2n}	$p(x_{2n})$	$q(x_{2n})$	$p(x_{2n}) \times q(x_{2n})$

1 multiplication per evaluation point!

$O(n)$ operations

Requirements

* Evaluation on canonical set of points
  ↳ same set of points for all p, q

Key Observation: Multiplication is EASY in Point-Value Representation

	$\hat{p}$	$\hat{q}$	$\widehat{p \cdot q}$
x_0	$p(x_0)$	$q(x_0)$	$p(x_0) \times q(x_0)$
x_1	$p(x_1)$	$q(x_1)$	$p(x_1) \times q(x_1)$
x_2	$p(x_2)$	$q(x_2)$	$p(x_2) \times q(x_2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_{2n-1}	$p(x_{2n-1})$	$q(x_{2n-1})$	$p(x_{2n-1}) \times q(x_{2n-1})$
x_{2n}	$p(x_{2n})$	$q(x_{2n})$	$p(x_{2n}) \times q(x_{2n})$

1 multiplication per evaluation point!

$O(n)$ operations

Requirements

* Evaluation on canonical set of points
  ↳ same set of points for all p, q
* Multiplication of $y_i \times z_i$ } Count as $O(1)$

Polynomial Multiplication Road Map

p, q in Coefficient Rep. $\xrightarrow{\text{polynomial multiplication}}$ $p \cdot q$ in Coefficient Rep

Polynomial Multiplication Road Map

$$\begin{tikzcd}
p, q \text{ in Coefficient Rep.} \arrow[r, dashed, "\text{polynomial multiplication}"] \arrow[d, "\text{Evaluation}"] & p \cdot q \text{ in Coefficient Rep} \\
\hat{p}, \hat{q} \text{ in Point-Value Rep} \arrow[r, "\text{FAST PV Multiply}"] & \widehat{p \cdot q} \text{ in Point-Value Rep} \arrow[u, "\text{Interpolation}"']
\end{tikzcd}$$

$$\begin{array}{llll}
x_0 & p(x_0) & \times\ q(x_0) & = \\
x_1 & p(x_1) & \times\ q(x_1) & = \\
\vdots & \vdots & \quad\ \vdots & \\
x_{n-1} & p(x_{n-1}) & \times\ q(x_{n-1}) & = \\
x_n & p(x_n) & \times\ q(x_n) & =
\end{array}$$

Announcements

* HW4 Ongoing.
* <u>Looking Ahead</u>: Prelim #2 on 27 March.
  - ↳ Only University-Approved Conflicts can be accommodated
  - ↳ Survey will come out in ~1 week.

Polynomial Multiplication Road Map

$$
\begin{tikzcd}
p, q \text{ in Coefficient Rep.} \arrow[r, dashed, "\text{polynomial multiplication}"] \arrow[d, "\text{Evaluation}"] & p \cdot q \text{ in Coefficient Rep} \\
\hat{p}, \hat{q} \text{ in Point-Value Rep} \arrow[r, "\text{FAST PV Multiply}"] & \widehat{p \cdot q} \text{ in Point-Value Rep} \arrow[u, "\text{Interpolation}"']
\end{tikzcd}
$$

x_0 $\quad p(x_0) \times q(x_0) =$

x_1 $\quad p(x_1) \times q(x_1) =$

$\vdots \quad \vdots \quad \vdots$

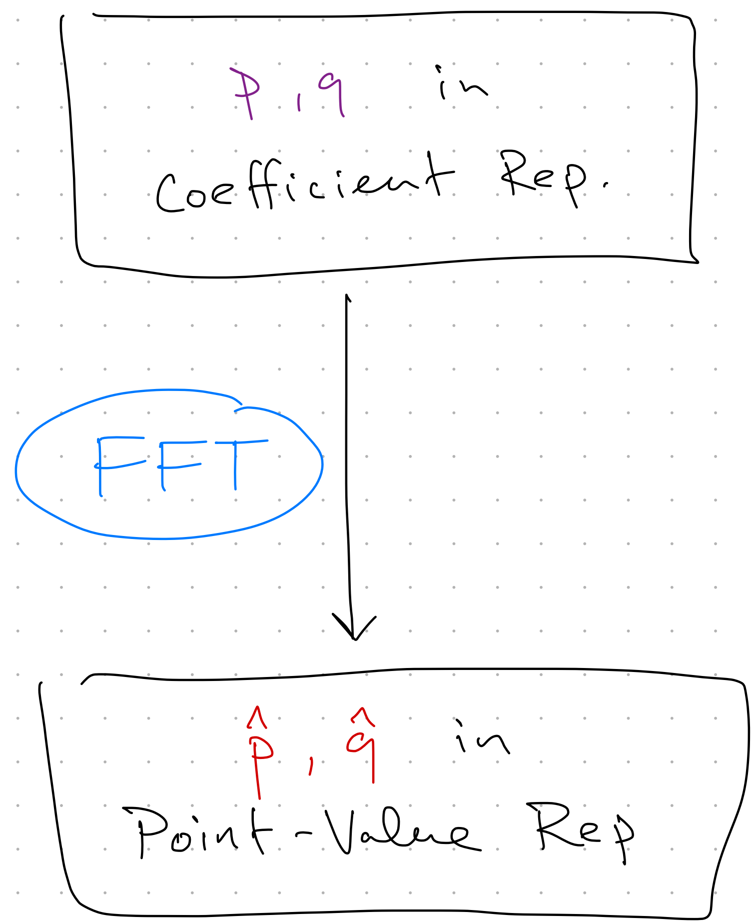
x_{n-1} $\quad p(x_{n-1}) \times q(x_{n-1}) =$

x_n $\quad p(x_n) \times q(x_n) =$

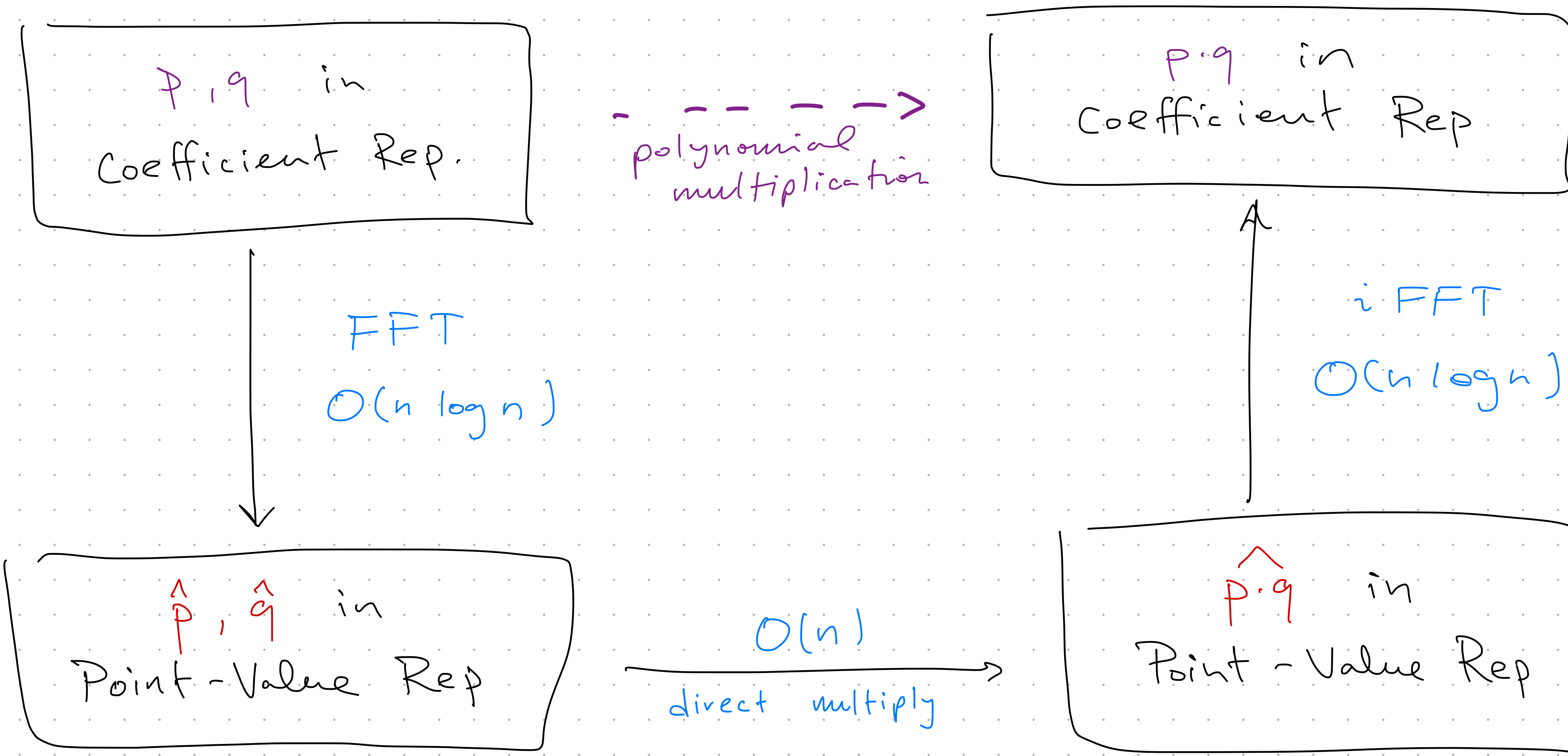
The Fast Fourier Transform.

Theorem. There exists an algo, FFT, that takes a degree-n polynomial $p(x)$ in coefficient rep, and returns the polynomial in Point-Value Rep, evaluated on a canonical set of points $x_0, x_1, \dots, x_n$ s.t. $\hat{p} = p(x_0), p(x_1) \dots p(x_n)$

using $O(n \log n)$ basic operations $(+, *)$



Polynomial Multiplication



<u>Corollary</u>. There exists an $O(n \log n)$ algorithm for multiplying degree-n polynomials.

p, q in Coefficient Rep.

$\Big\downarrow$ Evaluation

$\hat{p}, \hat{q}$ in Point-Value Rep

How quickly can we convert from

COEFFICIENT REPR.

$\Big\downarrow$ to

POINT-VALUE REPR. ?

Evaluation Problem.

Given: degree-n polynomial P

in coefficient representation

$P = P_n, P_{n-1}, \ldots P_1, P_0$ s.t. $p(x) = \sum_{i=0}^{n} P_i \cdot x^i$

Return: degree-n polynomial $\hat{P}$

in point-value (PV) representation

Evaluation Problem

<u>Given</u>: degree-n polynomial P
in <u>coefficient representation</u>

$$P = P_n, P_{n-1}, \ldots P_1, P_0 \quad \text{s.t.} \quad p(x) = \sum_{i=0}^{n} P_i \cdot x^i$$

<u>Return</u>: degree-n polynomial $\hat{p}$
in <u>point-value (PV) representation</u>

That is:

<u>canonical</u> set of points $x_0, x_1, \ldots, x_{n-1}, x_n$

with evaluations $\underbrace{p(x_0), p(x_1), \ldots, p(x_{n-1}), p(x_n)}_{\hat{p}}$

Naive Evaluation

* Choose $n+1$ points $x_0, x_1, \ldots, x_{n-1}, x_n$

* For each $j = 0, 1, \ldots, n$

  Compute $\hat{P}_j \leftarrow p(x_j) = \underbrace{\sum_{i=0}^{d} p_i \cdot x_j^i}_{\Omega(d) \text{ operations}}$

Naive Evaluation

* Choose $n+1$ points $x_0, x_1, \ldots, x_{n-1}, x_n$

* For each $j = 0, 1, \ldots n$

  Compute $\hat{P}_j \leftarrow p(x_j) = \underbrace{\sum_{i=0}^{d} p_i \cdot x_j^i}$

  $\Omega(d)$ operations

RT.

* $d+1$ evaluations
* $\Omega(d)$ operations per evaluation

$\Rightarrow \Omega(d^2)$ evaluation time.

Evaluation Problem.

Given: degree-n polynomial p

in coefficient representation

$$p = p_n, p_{n-1}, \ldots, p_1, p_0 \quad \text{s.t.} \quad p(x) = \sum_{i=0}^{n} p_i \cdot x^i$$

Return: degree-n polynomial $\hat{p}$

in point-value (PV) representation

which points?

That is:

canonical set of points $x_0, x_1, \ldots, x_{n-1}, x_n$

with evaluations $\underbrace{p(x_0), p(x_1), \ldots, p(x_{n-1}), p(x_n)}_{\hat{p}}$

By choosing points carefully, can we save in Divide & Recurse?

<u>Idea</u>. Look for points x_0, x_1 s.t. $p(x_0) = p(x_1)$

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

$$p(x) = 7x^6 + 3x^4 - x^2 + 5$$

By choosing points carefully,
can we save in Divide & Recurse?

<u>Idea</u>. Look for points x_0, x_1 s.t. $p(x_0) = p(x_1)$

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

$$\begin{matrix} x_0 = 1 \\ x_1 = -1 \end{matrix} \implies p(x_0) = p(x_1) = 1$$

$$p(x) = 7x^6 + 3x^4 - x^2 + 5$$

only even degrees

$$\begin{matrix} x_0 = 1 \\ x_1 = -1 \end{matrix} \implies p(x_0) = p(x_1)$$

By choosing points carefully, can we save in Divide & Recurse?

<u>Idea</u>. Look for points x_0, x_1 s.t. $p(x_0) \neq p(x_1)$

$p(x_0) \Rightarrow$ value of $p(x_1)$

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

$x_0 = 1$
$\Longrightarrow$
$x_1 = -1$

$$p(x) = 6x^5 + 3x^3 - x$$

$x_0 = 1$
$\Longrightarrow$
$x_1 = -1$

By choosing points carefully,

can we save in Divide & Recurse?

Idea. Look for points x_0, x_1 s.t. $p(x_0) \neq p(x_1)$

$p(x_0) \Rightarrow$ value of $p(x_1)$

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

$x_0 = 1$

$x_1 = -1$ $\Longrightarrow$ $p(x_0) = -p(x_1) = 1$

$$p(x) = 6x^5 + 3x^3 - x$$

only odd degrees

$x_0 = 1$

$x_1 = -1$ $\Longrightarrow$ $p(x_0) = -p(x_1)$

Idea: Divide & Recurse Based on degree

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

$$= P_{even}(x^2) + x \cdot P_{odd}(x^2)$$

$$P_{even}(x^2) = 2x^2 - 1$$

$$P_{odd}(x^2) = 3$$

Idea. Divide & Recurse Based on degree

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

$$= P_{even}(x^2) + x \cdot P_{odd}(x^2)$$

$P_{even}(x^2) = 2x^2 - 1$

$P_{odd}(x^2) = 3$

$$P_{even}(y) = 2y - 1 = \sum_{i=0}^{n/2} P_{2i} \cdot y^i$$

$$P_{odd}(y) = 3 = \sum_{i=0}^{n/2-1} P_{2i+1} \cdot y^i$$

Idea: Divide & Recurse Based on degree

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

$$= \underline{P_{even}(x^2) + x \cdot P_{odd}(x^2)}$$

$$P_{even}(x^2) = 2x^2 - 1 \qquad P_{odd}(x^2) = 3$$

$$\Downarrow$$

$$p(1) = P_{even}(1) + P_{odd}(1)$$

$$p(-1) = P_{even}(1) - P_{odd}(1)$$

To evaluate $p(1)$ and $p(-1)$ only need recursive calls on $x^2 = 1$.

Idea: Divide & Recurse Based on degree

$$p(1) = P_{even}(1) + P_{odd}(1)$$

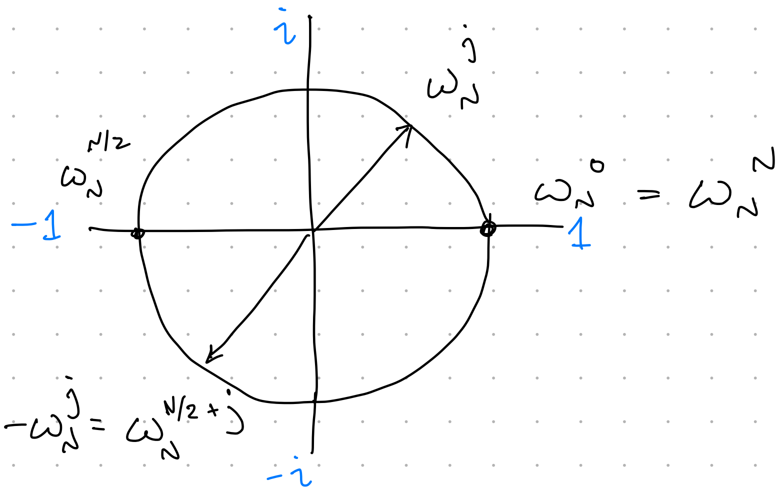
$$p(-1) = P_{even}(1) - P_{odd}(1)$$

To evaluate $p(1)$ and $p(-1)$ only need recursive calls on $x^2 = 1$.

Need $x_i^2 = x_j^2$ structure to hold recursively!

To evaluate $p(x_i)$ and $p(x_j)$ only need recursive calls on $x_i^2 = x_j^2$.

Evaluate on "Roots of Unity"



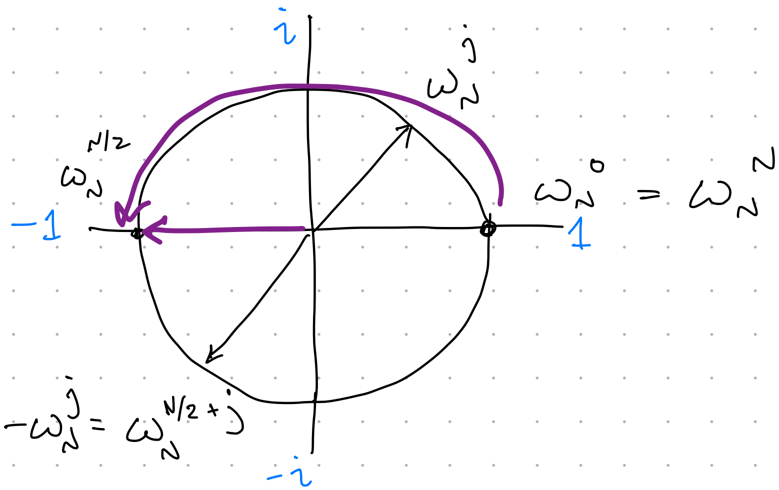
$\omega_N \equiv N^{th}$ root of unity

Complex Number s.t. $\omega_N^N = 1$

$$\vec{\omega}_N = \omega_N^0 \ \omega_N^1 \ \omega_N^2 \ \cdots \ \omega_N^{n-2} \ \omega_N^{n-1} \ \omega_N^n$$

For $N = 2^d > n$

Evaluate on "Roots of Unity"



$\omega_N \equiv N^{th}$ root of unity

Complex Number s.t. $\omega_N^N = 1$

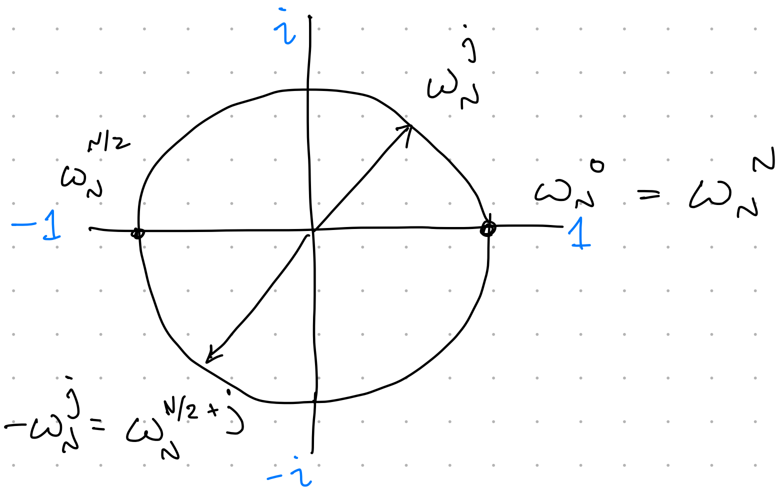
$$\vec{\omega}_N = \omega_N^0 \; \omega_N^1 \; \omega_N^2 \; \cdots \; \omega_N^{n-2} \; \omega_N^{n-1} \; \omega_N^n$$

Key properties.

(1) $\omega_N^{N/2} = -1 \implies (\omega_N^j)^2 = (\omega_N^{j+N/2})^2$

$\implies$ 1 recursive call, 2 evaluations!

Evaluate on "Roots of Unity"



$\omega_N \equiv N^{th}$ root of unity

Complex Number s.t. $\omega_N^N = 1$

$$\vec{\omega}_N = \omega_N^0 \; \omega_N^1 \; \omega_N^2 \; --- \; \omega_N^{n-2} \; \omega_N^{n-1} \; \omega_N^n$$

Key properties:

(1) $\omega_N^{N/2} = -1 \implies (\omega_N^j)^2 = (\omega_N^{j+N/2})^2$

$\implies$ 1 recursive call, 2 evaluations!

(2) $\omega_N^2 \equiv (N/2)^{th}$ root of unity

$\implies$ Root of unity structure preserved during Recursion

FFT : Algorithm for the Evaluation problem.

Given polynomial of degree-n, P

Returns $\hat{p}$, evaluation at

N^{th} Roots of Unity

$$\vec{\omega}_N = \omega_N^0 \quad \omega_N^1 \quad \cdots \quad \omega_N^{N-1}$$

for $N = 2^d > n$.

$$\mathrm{FFT}\left(p = p_n p_{n-1} p_{n-2} \cdots p_2 p_1 p_0,\ \omega_N\right)$$

Split p into

$$p_{even} = p_n p_{n-2} \cdots p_2 p_0$$

$$p_{odd} = p_{n-1} p_{n-3} \cdots p_3 p_1$$

// Coefficients of even & odd degree monomials

Goal:

$$p(x) = p_{even}(x^2) + x \cdot p_{odd}(x^2)$$

$$\text{FFT}\left(p = p_n p_{n-1} p_{n-2} \text{ --- } p_2 p_1 p_0,\ \omega_N\right)$$

Split p into

$$P_{even} = p_n p_{n-2} \text{ --- } p_2 p_0$$

$$P_{odd} = p_{n-1} p_{n-3} \text{ --- } p_3 p_1$$

// Coefficients of even & odd degree monomials

$$\hat{P}_{even} \leftarrow \text{FFT}\left(P_{even},\ \omega_N^2\right)$$

$$\hat{P}_{odd} \leftarrow \text{FFT}\left(P_{odd},\ \omega_N^2\right)$$

// Evaluate P_{even} and P_{odd} on $(N/2)^{th}$ Roots

BY Key Property (2)

Goal:

$$p(x) = P_{even}(x^2) + x \cdot P_{odd}(x^2)$$

$$\mathrm{FFT}\left(p = p_n p_{n-1} p_{n-2} \cdots p_2 p_1 p_0,\ \omega_N\right)$$

Split p into

$$p_{even} = p_n p_{n-2} \cdots p_2 p_0$$

$$p_{odd} = p_{n-1} p_{n-3} \cdots p_3 p_1$$

// Coefficients of even & odd degree monomials

$$\hat{p}_{even} \leftarrow \mathrm{FFT}\left(p_{even},\ \omega_N^2\right)$$

$$\hat{p}_{odd} \leftarrow \mathrm{FFT}\left(p_{odd},\ \omega_N^2\right)$$

// Evaluate p_{even} and p_{odd} on $(N/2)^{th}$ Roots

For $j = 0, \ldots, N$.

$$\hat{p}_j = \underbrace{\left(\hat{p}_{even}\right)_{j \bmod N/2}}_{p_{even}\left((\omega_N^j)^2\right) = p_{even}\left((\omega_N^{j+N/2})^2\right)} + \omega_N^j \cdot \left(\hat{p}_{odd}\right)_{j \bmod N/2}$$

BY Key Property (1)

$$\mathrm{FFT}\left(p = p_n p_{n-1} p_{n-2} \cdots p_2 p_1 p_0,\ \omega_N\right)$$

Split p into

$P_{even} = p_n p_{n-2} \cdots p_2 p_0$

$P_{odd} = p_{n-1} p_{n-3} \cdots p_3 p_1$

// Coefficients of even & odd degree monomials

$\hat{P}_{even} \leftarrow \mathrm{FFT}(P_{even}, \omega_N^2)$

$\hat{P}_{odd} \leftarrow \mathrm{FFT}(P_{odd}, \omega_N^2)$

// Evaluate P_{even} and P_{odd} on $(N/2)^{th}$ Roots

For $j = 0, \ldots, N$.

$$\hat{p}_j = \left(\hat{P}_{even}\right)_{j \bmod N/2} + \omega_N^j \cdot \left(\hat{P}_{odd}\right)_{j \bmod N/2}$$

Return $\hat{p}$.

// $p(x) = P_{even}(x^2) + x \cdot P_{odd}(x^2)$

where $(\omega_N^j)^2 = (\omega_N^{j+N/2})^2$

For $N = 2^d > n$

$\text{FFT}\left(p = p_n p_{n-1} p_{n-2} \cdots p_2 p_1 p_0,\ \omega_N\right)$

if $|p| = 1$ Return $\hat{p}_0 = p_0$ // Base case, constant deg 0 polynomial

Split p into

$p_{even} = p_n p_{n-2} \cdots p_2 p_0$ // Coefficients of even & odd degree monomials

$p_{odd} = p_{n-1} p_{n-3} \cdots p_3 p_1$

$\hat{p}_{even} \leftarrow \text{FFT}\left(p_{even},\ \omega_N^2\right)$ // Evaluate p_{even} and p_{odd} on $(N/2)^{th}$ Roots

$\hat{p}_{odd} \leftarrow \text{FFT}\left(p_{odd},\ \omega_N^2\right)$

For $j = 0, \ldots, N$.

$$\hat{p}_j = \left(\hat{p}_{even}\right)_{j \bmod N/2} + \omega_N^j \cdot \left(\hat{p}_{odd}\right)_{j \bmod N/2}$$

Return $\hat{p}$.

// $p(x) = p_{even}(x^2) + x \cdot p_{odd}(x^2)$

where $(\omega_N^j)^2 = (\omega_N^{j+N/2})^2$

Running Time?

* Each FFT makes 2 Recursive Calls,
* Each of size $N/2$
* Linear post-processing operations

$$T(N) \leq 2 \cdot T(N/2) + c \cdot N \quad \text{operations}$$

choose $N = \Theta(n)$

$\Rightarrow O(n \log n)$ basic operations