CS 170

DISCUSSION 2!

AGENDA

1. FFT Review/Practice

WHAT IS FFT?

- Multiplying polynomials efficiently!

$p = 3x^2 + 2x + 3$ $\qquad$ $q = 5x + 4$

$$p \cdot q = (3x^2 + 2x + 3) \cdot (5x + 4)$$
$$= 3x^2 \cdot (5x+4) + 2x \cdot (5x+4) + 3 \cdot (5x+4)$$
$$r(x) = 15x^3 + 22x^2 + 23x + 12$$

> Takes $O(mn)$ time! Or $O(n^2)$
(if p and q) have the same degree.

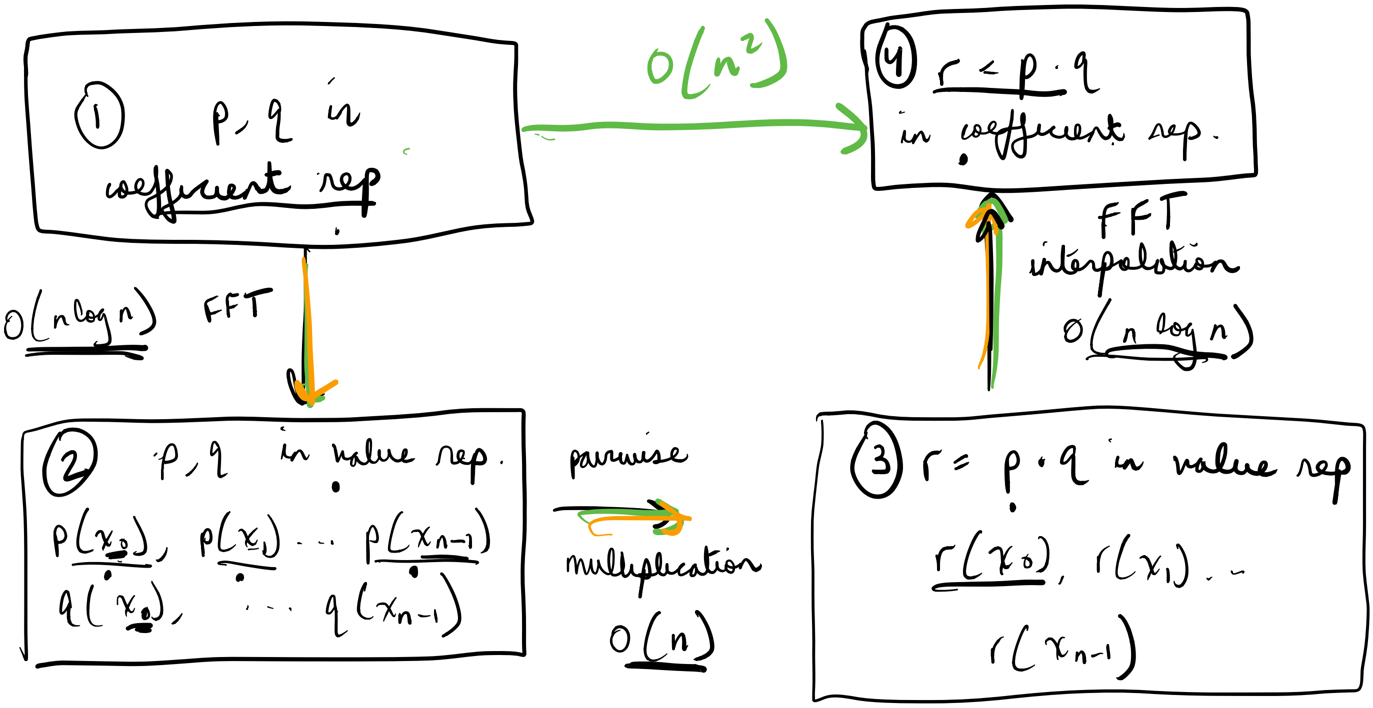
POLYNOMIAL REPRESENATIONS

$p = \underline{a}x + \underline{b}$

Coefficient form	Value form
$p(x) = \underline{3}x^{\underline{2}} + \underline{2}x + \underline{3}$	$p(0) = 3$ $p(1) = 8$ $p(2) = 19$
$q(x) = 5x + 4$	$q(0) = 4$ $q(1) = 9$

n+1 values, where n is the degree of the polynomial

INTERPOLATION OVERVIEW

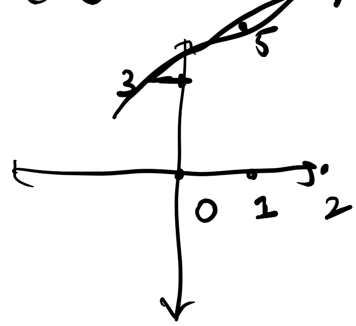


> FFT is an interpolation algorithm!

- Go from coefficient representation to value representation (using specific values).

FFT DIVIDE & CONQUER

$f(x) = 2x+3$



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

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

$$= 5x^4 + x^2 - 2 \;|\; + 3x^3 + x$$

$$= (5x^2 + x - 2)[x^2] + x(3x + 1)[x^2]$$

even coefficients, $E(x)$

evaluated at

odd coefficients, $O(x)$

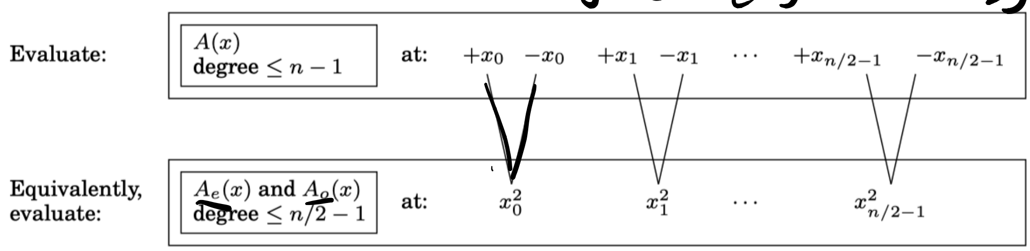
What if we want to find $p(x)$ and $p(-x)$?

$$p(x) = E(x^2) + x \cdot O(x^2)$$
$$p(-x) = \underline{E(x^2)} - x \cdot \underline{O(x^2)}$$

We can reuse the $E(x^2)$ and $O(x^2)$ calculation for both!

If we want to evaluate $p(x)$ at n paired points $\underline{\pm x_0, \ldots \pm x_{\frac{n}{2}-1}}$, then we need to evaluate $\underline{E(x)}$ and $\underline{O(x)}$ at $\frac{n}{2}$ points, $\underline{x_0^2, \ldots x_{\frac{n}{2}-1}^2}$.

$$T(n) = O(n \log n) \leftarrow T(n) = 2T\left(\tfrac{n}{2}\right) + O(n)$$



n

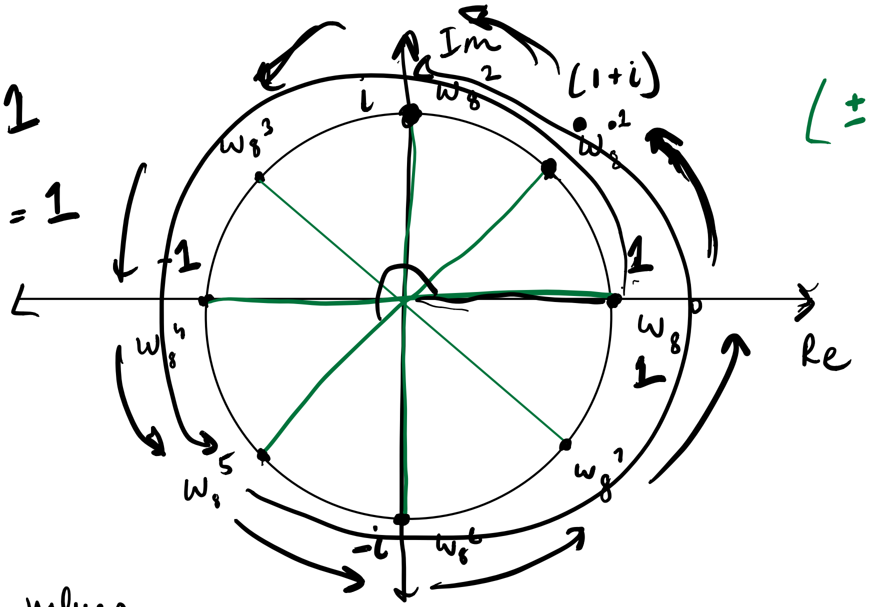
$n/2$

Only catch – need to make sure that $x_0^2, \ldots x_{\frac{n}{2}-1}^2$ are $\pm$ pairs too! so we can solve for $E(x_0^2), \ldots E(x_{\frac{n}{2}-1})^2$ and $O(x_0^2), \ldots O(x_{\frac{n}{2}-1})^2$ recursively.

So we use roots of unity!

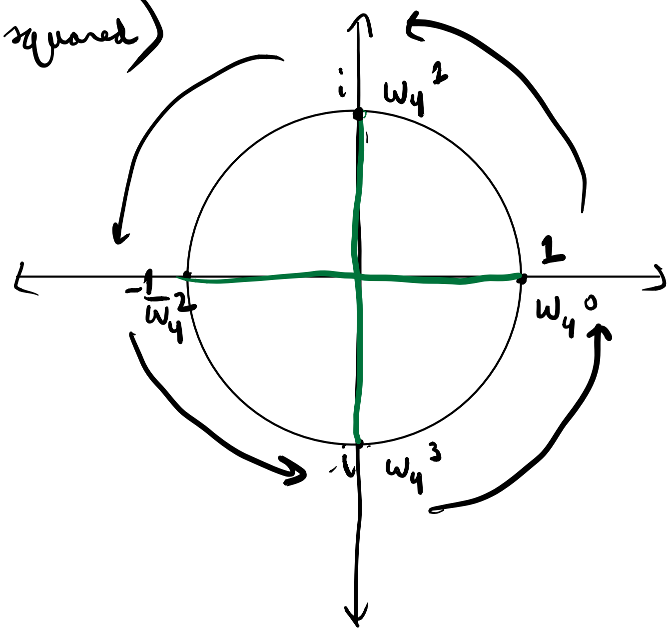
$w_8^8 = 1$

$(w_8^2)^8 = 1$



(± pairs are connected in green)

(when values are squared)



(still ± pairs)!

$w_8^8 = 1$

$w_8 = w_8^2$

w_n

> Important note : $\underline{w_8^2} = \underline{w_4}$

In general : $w_{2n}^2 = w_n$

Practice

Fast Fourier Transform! The *Fast Fourier Transform* FFT(p, n) takes arguments n, some power of 2, and p is some vector $[p_0, p_1, \ldots, p_{n-1}]$.

Treating p as a polynomial $P(x) = p_0 + p_1 x + \ldots + p_{n-1}x^{n-1}$, the FFT computes the following matrix multiplication in $\mathcal{O}(n \log n)$ time:

$$\begin{bmatrix} P(1) \\ P(\omega_n) \\ P(\omega_n^2) \\ \vdots \\ P(\omega_n^{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^1 & \omega_n^2 & \cdots & \omega_n^{(n-1)} \\ 1 & \omega_n^2 & \omega_n^4 & \cdots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{(n-1)} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)(n-1)} \end{bmatrix} \cdot \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ \vdots \\ p_{n-1} \end{bmatrix}$$

If we let $E(x) = p_0 + p_2 x + \ldots p_{n-2}x^{n/2-1}$ and $O(x) = p_1 + p_3 x + \ldots p_{n-1}x^{n/2-1}$, then $P(x) = E(x^2) + xO(x^2)$, and then $FFT(p, n)$ can be expressed as a divide-and-conquer algorithm:

1. Compute $E' = \text{FFT}(E, n/2)$ and $O' = \text{FFT}(O, n/2)$.
2. For $i = 0 \ldots n-1$, assign $P(\omega_n^i) \leftarrow E((\omega_n^i)^2) + \omega_n^i O((\omega_n^i)^2)$

$$\begin{cases} p(x) = \boxed{E(x^2)} + x \cdot \boxed{O(x^2)} \\ p(-x) = E(x^2) - x \cdot O(x^2) \end{cases}$$

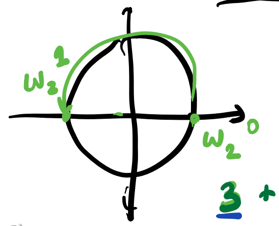
(a) Let $p = [p_0]$. What is FFT$(p, 1)$?

$[1, 4, 7]$ $\quad 7x^3 + 4x + 1$

p_0

$\omega_2^0 \quad \omega_2^1$

$[5, -3]$



$3 + x \cdot 2$

(b) Use the FFT algorithm to compute $\boxed{\text{FFT}([1,4],2)}$ and FFT$([3,2],2)$.

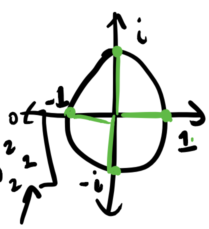
$x^2 = 1$

$(1, -1)$

$[1, 4] \Rightarrow 4x + 1$ $\qquad [3, 2] \Rightarrow \boxed{2}x + \boxed{3}$

$$\begin{array}{c|c|c} 1 & 3 + 1 \cdot 2 = \begin{bmatrix} 5 \\ 1 \end{bmatrix} & 1 + 1 \cdot 4 = 5 \\ -1 & 3 - 1 \cdot 2 = & 1 - 1 \cdot 4 = -3 \end{array}$$

(c) Use your answers to the previous parts to compute FFT$([1, 3, 4, 2], 4)$. $\quad 2x^3 + 4x^2 + 3x + 1$



$\begin{bmatrix} 5 \\ -3 \end{bmatrix} \begin{bmatrix} \omega_2^0 \\ \omega_2^1 \end{bmatrix}$

$[\, i, \ -1, \ -i, \ 1 \,]$

$$\begin{cases} f(x) = E(x^2) + x \cdot O(x^2) \\ f(-x) = E(x^2) - x \cdot O(x^2) \end{cases}$$

value -1

$\begin{bmatrix} \omega_4^1 \\ \omega_4^3 \end{bmatrix}$

$$\begin{array}{c|l|l} i & \text{FFT}([1,4],2) + i \cdot \text{FFT}([3,2],2) & -3 + i \cdot 1 \\ -i & \text{FFT}([1,4],2) - i \cdot \text{FFT}([3,2],2) & -3 - i \cdot 1 \\ 1 & \text{FFT}([1,4],2) + 1 \cdot \text{FFT}([3,2],2) & 5 + 1 \cdot 5 \\ -1 & \text{FFT}([1,4],2) - 1 \cdot \text{FFT}([3,2],2) & 5 - 1 \cdot 5 \end{array}$$

1 term

$FFT([1,4],2) = [5, -3]$

$1 \quad -1$

$FFT([3,2],2) = [5, 1]$

$1 \quad -1$

$w_2^0 \quad w_2^1$

$FFT([1,4], 2)$

$4x + 1$

$w_2^0 \to 5$

$w_2^1 \to -3$

$$T(n) = T\left(\frac{n}{10}\right) + T\left(\frac{9n}{10}\right) + O(n)$$

$O(n)$

$\log\left(\frac{10}{9}\right)$

$O\left(\frac{n}{10}\right) \longrightarrow O\left(\frac{9n}{10}\right) \quad O(n)$

2 Cubed Fourier

(a) Cubing the 9^{th} roots of unity gives the 3^{rd} roots of unity. Next to each of the third roots below, write down the corresponding 9^{th} roots which cube to it. The first has been filled for you. *We will use ω_9 to represent the primitive 9^{th} root of unity, and ω_3 to represent the primitive 3^{rd} root.*

$\omega_3^0 : \omega_9^0,$,

$\omega_3^1 :$, ,

$\omega_3^2 :$, ,

$\omega_3^2 :$, ,

(b) You want to run FFT on a degree-8 polynomial, but you don't like having to pad it with 0s to make the (degree+1) a power of 2. Instead, you realize that 9 is a power of 3, and you decide to work directly with 9th roots of unity and use the fact proven in part (a). Say that your polynomial looks like $P(x) = a_0 + a_1x + a_2x^2 + \ldots + a_8x^8$. Describe a way to split $P(x)$ into three pieces so that you can make an FFT-like divide-and-conquer algorithm.

3 Predicting a Weighted Average

You have a time-series dataset $y_0, y_1, \ldots, y_{n-1}$ where all $y_i \in \mathbb{R}$. You are given fixed coefficients $c_0, \ldots, c_{n-2}$, which give the following prediction for day $t \geq 1$:

$$p_t = \sum_{k=0}^{t-1} c_k y_{t-1-k}$$

You would like to evaluate the accuracy of this prediction on the dataset by computing the *mean squared error*, given by

$$\frac{1}{n-1} \sum_{t=1}^{n-1} (p_t - y_t)^2$$

Find an $\mathcal{O}(n \log n)$ time algorithm to compute the mean squared error, given dataset $y_0, y_1, \ldots, y_{n-1}$ and coefficients $c_0, \ldots, c_{n-2}$.

Hint: Recall that if $p(x) = p_0 + p_1 x + p_2 x^2 + \ldots p_{n-1} x^{n-1}$ and $q(x) = q_0 + q_1 x + q_2 x^2 + \ldots q_{n-1} x^{n-1}$, then their product is $p(x) \cdot q(x) = r(x) = r_0 + r_1 x + \ldots + r_{2n-2} x^{2n-2}$, where

$$r_j = \sum_{k=0}^{j} p_k q_{j-k}$$