Note: Your TA may not get to all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. The discussion worksheet is also a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

1 FFT Intro

We will use ω_n to denote the first n-th root of unity $\omega_n = e^{2\pi i/n}$. The most important fact about roots of unity for our purposes is that the squares of the $2n$-th roots of unity *are* the n-th roots of unity.

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

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 $i = 0 \dots n-1$, assign $P(\omega_n^i) \leftarrow E((\omega_n^i)^2) + \omega_n^i O((\omega_n^i)^2)$

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

$[P(1)] = [1] \rightarrow$ return $[1]$

(b) Use the FFT algorithm to compute FFT$([1, 4], 2)$ and FFT$([3, 2], 2)$. $\rightarrow [5, 1]$

$FFT([1], 1) = 1 \qquad FFT([4], 1) = 4$

$P(W_2^0) = P(1) \qquad P(W_2^1) = P(e^{\pi i}) = P(-1)$

$[5, -3] \qquad \overset{E(1)\;\;E(-1)}{E' = [5, -3]} \qquad \overset{O(1)\;\;O(-1)}{O' = [5, 1]}$

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

$P(1) \quad 1 \quad e^{2\pi i/4}$

$E' = 5 - 3x \qquad O' = 5 + x$

$P(i) \quad -1$

$P(-1) \quad 1$

$P(-i) \quad 1$

$P(1) = E(1) + O(1) = 10$

$P(i) = E(-1) + i * O(-1) = -3 + i$

$P(1) = E(1) - O(1) = 0$

$P(-i) = E(-1) - i * O(-1) = -3 - i$

(d) Describe how to multiply two polynomials $p(x), q(x)$ in coefficient form of degree at most d.

FFT alogrithm

$d+1$ extends to $= 2^n$ padding

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.*

$\omega_3^0 : \omega_9^0, w_9^3, w_9^6$

$\omega_3^1 : w_9^1, w_9^4, w_9^7$

$\omega_3^2 : w_9^2, w_9^5, w_9^8$

(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.

$E(x) = a_0 + a_3 x + a_6 x^2$ $\quad E' = FFT(E, 3)$

$O(x) = a_1 + a_4 x + a_7 x^2$ $\quad O' = FFT(O, 3)$

$R(x) = a_2 + a_5 x + a_8 x^2$ $\quad R' = FFT(R, 3)$

$P(x) = E(x^3) + x\,O(x^3) + x^2 R(x^3)$

$P(w_n^i) = E(w_n^{i3}) + w_n^i\, O(w_n^{i3}) + (w_n^i)^2 R(w_n^{i3})$

$T(N) = 3T(\frac{N}{3}) + O(N)$

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

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