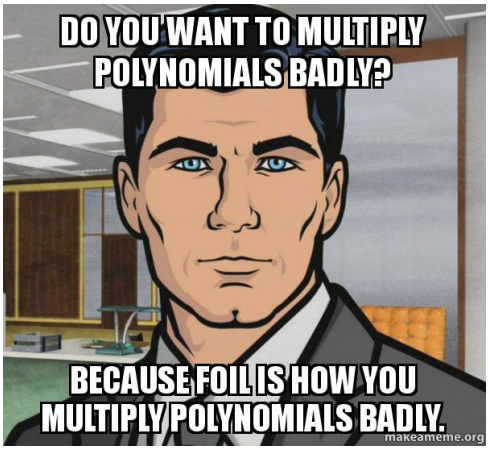
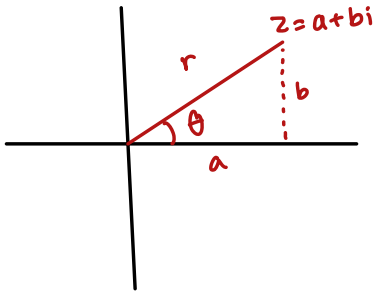
Discussion 2: FFT

feedback form: bit.ly/cindy-feedback-cs170



ROOTS OF UNITY

Complex Numbers:



$z = a + bi$ (rectangular)

(a: real, bi: imaginary)

$z = r(\cos\theta + i\sin\theta)$ (polar)

$a = r\cos\theta$

$b = r\sin\theta$

Using Euler's Formula:

$$re^{i\theta} = r(\cos\theta + i\sin\theta)$$

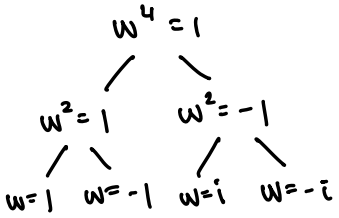
n^{th} roots of unity: n complex numbers satisfying $w^n = 1$

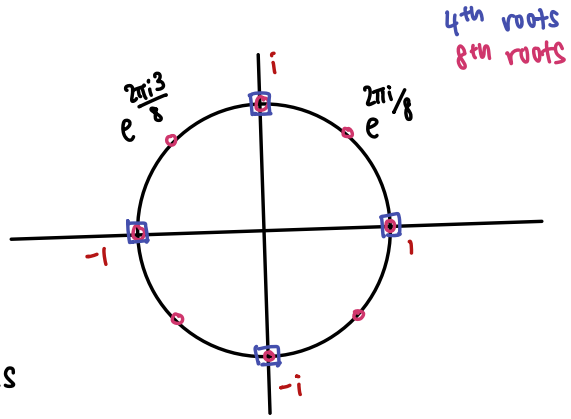
$$w_k = e^{2\pi i k/n} \qquad k = 0, 1, 2, \ldots, n-1$$

ex: what are the second roots of unity?

$w^2 = 1$ $\quad w = +1, -1$

fourth roots of unity?





Note: squaring n^{th} roots of unity equals $(\frac{n}{2})^{th}$ roots of unity

$$\left(e^{\frac{2\pi i 3}{8}}\right)^2 = e^{\frac{2\pi i 6}{8}} = e^{\frac{2\pi i 3}{4}}$$

* For n^{th} roots of unity, place n points evenly on unit circle

Note: Your TA probably will not cover all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. They are deliberately made long so they can serve as a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

1 Complex numbers review

A *complex number* is a number that can be written in the rectangular form $a + bi$ (i is the imaginary unit, with $i^2 = -1$). The following famous equation (*Euler's formula*) relates the polar form of complex numbers to the rectangular form:

$$re^{i\theta} = r(\cos\theta + i\sin\theta)$$

In polar form, $r \geq 0$ represents the distance of the complex number from 0, and θ represents its angle. Note that since $\sin(\theta) = \sin(\theta + 2\pi), \cos(\theta) = \cos(\theta + 2\pi)$, we have $re^{i\theta} = re^{i(\theta+2\pi)}$ for any r, θ.

The n-th *roots of unity* are the n complex numbers satisfying $\omega^n = 1$. They are given by

$$\omega_k = e^{2\pi ik/n}, \qquad k = 0, 1, 2, \ldots, n-1$$

(a) Let $x = e^{2\pi i3/10}, y = e^{2\pi i5/10}$ which are two 10-th roots of unity. Compute the product $x \cdot y$. Is this an n-th root of unity for some n? Is it a 10-th root of unity?

What happens if $x = e^{2\pi i6/10}, y = e^{2\pi i7/10}$?

$xy = \exp(2\pi i 3/10 + 2\pi i 5/10) = \exp\left(\frac{2\pi i 8}{10}\right) = \exp\left(\frac{2\pi i 4}{5}\right)$

10th root of unity and 5th root of unity.

$\exp\left(\frac{2\pi i 13}{10}\right) = \exp\left(\frac{2\pi i 3}{10}\right)$ "wind around"

(b) Show that for any n-th root of unity $\omega \neq 1$, $\sum_{k=0}^{n-1} \omega^k = 0$, when $n > 1$.

Hint: Use the formula for the sum of a geometric series $\sum_{k=0}^{n} \alpha^k = \frac{\alpha^{n+1}-1}{\alpha-1}$. It works for complex numbers too!

$\omega^n = 1$

$$\sum_{k=0}^{n-1} \omega^k = \frac{\omega^n - 1}{\omega - 1} = \frac{1-1}{\omega - 1} = 0$$

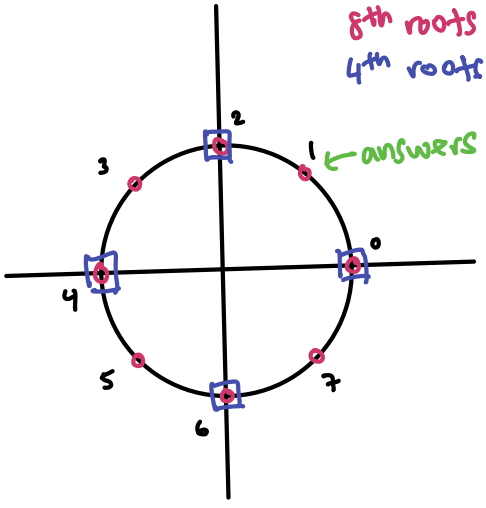
(c) (i) Find all ω such that $\omega^2 = -1$.

$\omega = +i, -i$

(ii) Find all ω such that $\omega^4 = -1$.

$\omega^8 = 1$, 8th roots of unity not 4th roots

$\omega = e^{2\pi i/8}, e^{2\pi i3/8}, e^{2\pi i5/8}, e^{2\pi i7/8}$



FAST FOURIER TRANSFORM

$$F(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_d x^d$$

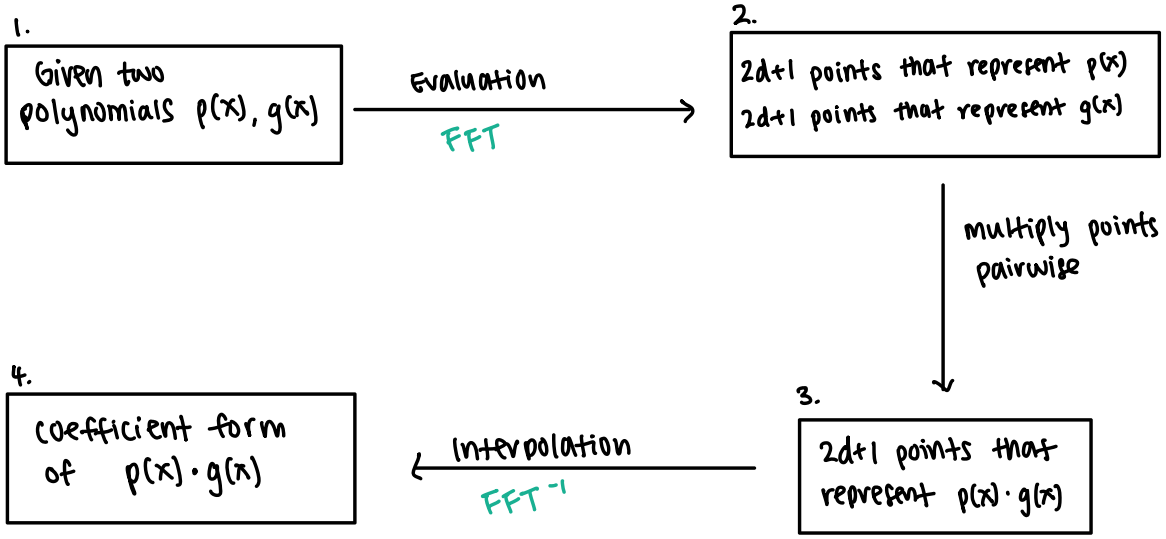
Coefficient representation
$a_0, a_1, \ldots a_d$

Evaluation →
← Interpolation

Value representation
$F(v_0) \ldots F(v_d)$

* d degree polynomial represented by d+1 distinct points

To multiply $p(x) \cdot g(x)$ of degree d:



Note: If the number of coefficients of a polynomial is not a power of 2, zero pad the polynomial.

$$x^2 + 2x + 7 \longrightarrow 0x^3 + x^2 + 2x + 7$$

FFT Algorithm:

$$P(x) = p_0 + p_1 x + \ldots + p_{n-1} x^{n-1}$$

even terms: $p_0, p_2 x^2, p_4 x^4, \ldots, p_{n-2} x^{n-2}$

odd terms: $p_1 x, p_3 x^3, p_5 x^5, \ldots, p_{n-1} x^{n-1}$

$$P(x) = E(x^2) + xO(x^2)$$

$$E(x) = p_0 + p_2 x + p_4 x^2 + \ldots + p_{n-2} x^{n/2-1}$$

$$O(x) = p_1 + p_3 x + p_5 x^2 + \ldots + p_{n-1} x^{n/2-1}$$

* wlog assume n is a power of 2

Divide and Conquer:

1. Compute $E(x)$ and $O(x)$
2. For $i = 0 \ldots n-1$ assign $P(w_n^i) = E((w_n^i)^2) + w_n^i O((w_n^i)^2)$

This is interpolation! The n points are the n^{th} roots of unity.

Given paired points of $\pm x$:

$$P(-x) = E(x^2) - xO(x^2) \qquad P(x) = E(x^2) + xO(x^2)$$

* Evaluating $P(x)$ at n paired points $\pm x_0, \pm x_1, \ldots, \pm x_{n/2-1}$ reduces to evaluating $E(x_i^2)$ and $O(x_i^2)$. Subproblem is half the size.

$$T(n) = 2T(n/2) + O(n) \Rightarrow O(n \log n)$$

2 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* $\text{FFT}(p, n)$ 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 value of $P(x)$ for all x that are n-th roots of unity by doing the following matrix multiplication in $\mathcal{O}(n \log n)$ time:

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

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 $P(\omega_n^i) \leftarrow E((\omega_n^i)^2) + \omega_n^i O((\omega_n^i)^2)$

Also observe that:

$$\frac{1}{n}\begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega_n^{-1} & \omega_n^{-2} & \ldots & \omega_n^{-(n-1)} \\ 1 & \omega_n^{-2} & \omega_n^{-4} & \ldots & \omega_n^{-2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{-(n-1)} & \omega_n^{-2(n-1)} & \ldots & \omega_n^{-(n-1)(n-1)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega_n^{1} & \omega_n^{2} & \ldots & \omega_n^{(n-1)} \\ 1 & \omega_n^{2} & \omega_n^{4} & \ldots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{(n-1)} & \omega_n^{2(n-1)} & \ldots & \omega_n^{(n-1)(n-1)} \end{bmatrix}^{-1}$$

(You should verify this on your own!) And so given the values $P(1), P(\omega_n), P(\omega_n^2) \ldots$, we can compute P by doing the following matrix multiplication:

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

This can be done in $O(n \log n)$ time using a similar divide and conquer algorithm.

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

$[p_0]$ base case

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

FFT([1],1) = [1] = E(x) = 1
FFT([4],1) = [4] = O(x) = 4
P(1) = E(1) + O(1) = 1 + 4 = 5
P(-1) = E(1) − O(1) = 1 − 4 = -3

FFT([3],1) = [3] = E
FFT([2],1) = [2] = O
$w_2 = \pm 1$
P(1) = E(1) + O(1) = 3 + 2 = 5
P(-1) = E(1) − O(1) = 3 − 2 = 1

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

↑ evens ↑ odds

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

$w_4 = 1, -1, i, -i$

$P(1) = E(1) + O(1) = 5 + 5 = 10$

$P(-1) = E(1) - O(1) = 5 - 5 = 0$

$P(i) = E(-1) + iO(-1) = -3 + i$

$P(-i) = E(-1) - iO(-1) = -3 - i$

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

1. Take FFT of both p and q
2. Multiply evaluations
3. Take inverse of FFT to get p·q in coefficient form

3 Cartesian Sum

Let A and B be two sets of integers in the range 0 to $10n$. The *Cartesian sum* of A and B is defined as

$$A + B = \{a + b \mid a \in A, b \in B\}$$

i.e. all sums of an element from A and an element with B. For example, $\{1, 3\} + \{2, 4\} = \{3, 5, 7\}$.

Note that the values of $A + B$ are integers in the range 0 to $20n$. Design an algorithm that finds the elements of $A + B$ in $\mathcal{O}(n \log n)$ time, which additionally tells you for each $c \in A + B$, *how many* pairs $a \in A, b \in B$ there are such that $a + b = c$.

Hint: Notice that $(x^1 + x^3) \cdot (x^2 + x^4) = x^3 + 2x^5 + x^7$

Key: powers of the polynomial are the elements in the Cartesian sum

encode A and B into polynomials and use FFT to multiply them

O(nlogn)

4 Cubed Roots of Unity

(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.*

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

$\omega_3^1 : \omega_9^1, \omega_9^4, \omega_9^7$

$\omega_3^2 : \omega_9^2, \omega_9^5, \omega_9^8$

(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 $P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_8 x^8$. Describe a way to split $P(x)$ into three pieces (instead of two) so that you can make an FFT-like divide-and-conquer algorithm.

(c) What is the runtime of FFT when we divide the polynomial into three pieces instead of two?

b) Divide into thirds

$$P(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + a_6 x^6 + a_7 x^7 + a_8 x^8$$

$$P(x) = P_1(x^3) + x P_2(x^3) + x^2 P_3(x^3)$$

$$P_1(x^3) = a_0 + a_3 x^3 + a_6 x^6$$

$$P_2(x^3) = a_1 + a_4 x^3 + a_7 x^6$$

$$P_3(x^3) = a_2 + a_5 x^3 + a_8 x^6$$

c) $T(n) = 3T(n/3) + O(n) = O(n \log n)$