12 Fast Fourier Transform

- Divide and Conquer: FFT
- Next: Dynamic Programming

12.1 Fast Fourier Transform

$$
\begin{aligned}
c(x) &= a(x) \cdot b(x) \\
&= \left(\sum_{j=0}^{n-1} a_j x^j \right) \left(\sum_{k=0}^{n-1} b_k x^k \right) \\
&= \sum_{j,k=0}^{n-1} a_j b_k x^{j+k} \qquad \text{substitute } l = j + k \\
&= \sum_{l=0}^{2n-2} \left(\sum_{j=0}^{l} a_j b_{l-j} \right) x^l
\end{aligned}
$$

Main Idea: use polynomial interpolation.
A degree d polynomial is uniquely specified by its values at any $d+1$ distinct points.

Strategy:

1. Evaluate $a(x)$ and $b(x)$ on $2n-1$ points.
2. Evaluate $c(x)$ on those points.
3. Reconstruct the coefficients from these data.

For (1), we compute $O(n)$ things, each of which takes $O(n)$ time to evaluate individually.

Consider evaluating a polynomial of degree $d-1$ at d points $x_0, x_1, \cdots, x_{d-1}$.

$$a(x_j) = a_0 + a_1 x_j + \cdots + a_{d-1} x_j^{d-1}$$

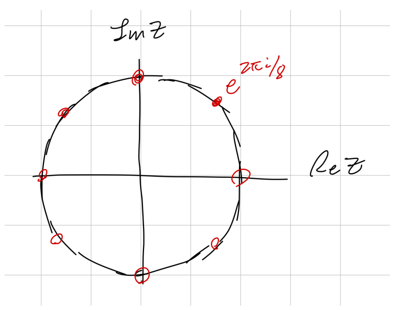
Assume that d is even.
Let

$$
\begin{aligned}
a_{\text{even}}(x) &= a_0 + a_2 x + a_4 x^2 + \cdots + a_{d-2} x^{\frac{d}{2}-1} \\
a_{\text{odd}}(x) &= a_1 + a_3 x + a_5 x^2 + \cdots + a_{d-1} x^{\frac{d}{2}-1}
\end{aligned}
$$

Then

$$a(x) = a_{\text{even}}(x^2) + x \cdot a_{\text{odd}}(x^2)$$

The natural choice for these points are the dth roots of unity



$$x_j = \omega_d^j \qquad \text{where} \quad w_d = e^{2\pi i/d}$$

$$x_j^2 = w_d^{2j} = w_d^{2j \mod d} = x_{2j \mod d}$$

Note that

$$e^{2\pi i} = 1$$

If $T(d)$ is the cost of evaluating $a(x)$ at x_j for $j \in \{0, 1, \cdots, d-1\}$, then we have $T(d) = 2T(d/2) + O(d) \implies T(d) = O(d \log d)$

For (3), consider how the coefficients of a polynomial relate to its evaluations at $\omega_d^0, \omega_d^1, \cdots, \omega_d^{d-1}$.

$$\begin{aligned} a(x) &= a_0 + a_1 x + a_2 x^2 + \cdots + a_{d-1} x^{d-1} \\ &= \begin{bmatrix} 1 & x & x^2 & \cdots & x^{d-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{d-1} \end{bmatrix} \end{aligned}$$

So,

$$\begin{bmatrix} a(\omega_d^0) \\ a(\omega_d^1) \\ a(\omega_d^2) \\ \vdots \\ a(\omega_d^{d-1}) \end{bmatrix} = \begin{bmatrix} 1 & \omega_d^0 & (\omega_d^0)^2 & \cdots & (\omega_d^0)^{d-1} \\ 1 & \omega_d^1 & (\omega_d^1)^2 & \cdots & (\omega_d^1)^{d-1} \\ 1 & \omega_d^2 & (\omega_d^2)^2 & \cdots & (\omega_d^2)^{d-1} \\ \vdots \\ 1 & \omega_d^{d-1} & (\omega_d^{d-1})^2 & \cdots & (\omega_d^{d-1})^{d-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{d-1} \end{bmatrix}$$

Where the middle matrix is called the discrete fourier transform.

F is a "unitary matrix": its inverse is easy to compute.
We have

$$x = \begin{bmatrix} a(\omega_d^0) \\ a(\omega_d^1) \\ a(\omega_d^2) \\ \vdots \\ a(\omega_d^{d-1}) \end{bmatrix}$$

We want

$$y = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{d-1} \end{bmatrix}$$

$$x = \frac{1}{\sqrt{d}} F y$$

$$\sqrt{d} F^{-1} x = F^{-1} F y = y$$

Because F is unitary, F^{-1} is ust like F, but with ω replaced by $\frac{1}{\omega}$