Divide and Conquer (Contd.)

→ Linear transforms:

$$\underset{m\times 1}{\bar{b}} = \underset{m\times n}{A}\,\underset{n\times 1}{\bar{v}}$$

$$\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}_{m\times n} \begin{bmatrix} \\ \\ \\ \end{bmatrix}_{n\times 1}.$$

↳ For each entry of $\bar{b}$, we have inner product of n length vectors.

Naive: $O(mn)$

$L \leftarrow$

$L_{ij} = \omega^{ij}$

ω is a primitive n^{th} root of unity

$\omega^k \neq 1 \;\; \forall\; k \in [1, n-1]$

$\omega = e^{\frac{2\pi i}{n}}$

$\omega^4 = 1$, $\omega^1 \neq 1$, $\omega^2 \neq 1$, $\omega^3 \neq 1$ } Primitive fourth root $\omega = -i, i$

If ω is primitive fourth root, then ω^2 is prim 2nd root.

$\{\alpha_1, \ldots, \alpha_n\}$

1-indexing for j

$$\underbrace{V(\alpha_1, \ldots, \alpha_n)}_{\text{Vandermonde matrices.}} = \begin{bmatrix} \alpha_1^0 & \alpha_1^1 & \cdots & \alpha_1^{n-1} \\ \vdots & & \alpha_i^{j-1} & \\ \alpha_n^0 & \alpha_n^1 & \cdots & \alpha_n^{n-1} \end{bmatrix}$$

$P(z) = \sum_{i=0}^{n-1} c_i z^i$

$P(\alpha_i) = \langle i^{th}$ row of $V(\bar{a}), \begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \end{bmatrix} \rangle$

$$\begin{bmatrix} P(\alpha_1) \\ \vdots \\ P(\alpha_n) \end{bmatrix} = V \cdot \begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \end{bmatrix} = \sum_{j=0}^{n-1} c_j \cdot \alpha_i^j$$

zero-indexing for j.

Given coefs of a polynomial, we are computing their evaluations.

Coefs ✓ Evaluations

$$\begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \end{bmatrix} = V^{-1} \begin{bmatrix} P(\alpha_1) \\ \vdots \\ P(\alpha_n) \end{bmatrix}$$

$$\begin{tikzcd} \text{Eval} \arrow[r, bend left, "V^{-1}"] & \text{Coefs} \arrow[l, bend left, "V"] \end{tikzcd}$$

We are interested in complexity of linear transformations when $\alpha_1, \dots, \alpha_n$ are $\omega^1, \omega^2, \dots, \omega^n$.

ω^{ij} $\qquad (L^{-1})_{ij} = \dfrac{\omega^{-ij}}{n} = \dfrac{\omega^{n-ij}}{n}$.

If transformation by L and L^{-1} were efficient (much better than $O(n^2)$) then we can do polynomial mult. efficiently.

$P(z) = \sum_{i=1}^{n} c_i z^{i-1}$ $\qquad Q(z) = \sum_{i=1}^{n} c_i' z^{i-1}$ $\qquad R(z) = P(z) \cdot Q(z)$

$\downarrow T(n)$ $\qquad T(n)$ $\qquad T'(n)$

$P(\omega), \dots, P(\omega^n)$ $\qquad Q(\omega), \dots, Q(\omega^n)$ $\qquad R(\omega), \dots, R(\omega^n)$

Caveat: R need not be a deg n poly.

$$\text{DFT}(n) \qquad \begin{array}{c|c} & j \\ \hline i & \begin{bmatrix} & & \\ & \omega^{ij} & \\ & & \end{bmatrix} \end{array}$$

primitive n^{th} root of unity.

$$\begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} & | & \\ - & \omega^{ij} & \\ & & \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$$

$$b_i = \sum_{j=1}^{n} \omega^{ij} \cdot a_j = \sum_{j=1}^{n/2} \omega^{ij} a_j + \underbrace{\sum_{j=\frac{n}{2}+1}^{n} \omega^{ij} \cdot a_j}$$

$$\sum_{j=\frac{n}{2}+1}^{n} \omega^{ij} a_j = \sum_{j'=1}^{n/2} \omega^{i(j'+\frac{n}{2})} \cdot a_{j'+\frac{n}{2}}$$

$j' = j - \frac{n}{2} \Rightarrow j = j' + \frac{n}{2}$ $$= \omega^{\frac{n}{2} i} \sum_{j'=1}^{n/2} \omega^{i \cdot j'} \cdot a_{j'+\frac{n}{2}}$$

$$b_i = \sum_{j=1}^{n/2} \omega^{ij} \cdot a_j + \omega^{\frac{n}{2} i} \cdot \sum_{j'=1}^{n/2} \omega^{ij'} \cdot a_{j'+\frac{n}{2}}$$

If $\omega^n = 1$, then what is $\omega^{\frac{n}{2}}$? $\omega^{n/2} = -1$.

$$b_i = \underbrace{\sum_{j=1}^{n/2} \omega^{ij} \cdot a_j} + (-1)^i \cdot \sum_{j'=1}^{n/2} \omega^{ij'} \cdot a_{j'+\frac{n}{2}}$$

Is this a $DFT_{\frac{n}{2}}$? <u>Ans</u>: No because ω is n^{th} root and not $\frac{n}{2}^{th}$ root.

<u>Obs</u>: ω^2 is $\frac{n}{2}^{th}$ primitive root of unity.

If i is even: Say $i = 2p$.

$$b_i = \sum_{j=1}^{n/2} (w^2)^{p \cdot j} \cdot a_j + \sum_{j'=1}^{n/2} (w^2)^{p \cdot j'} \cdot a_{j'+\frac{n}{2}}$$

j' is a notation and can be replaced with j.

$$= \sum_{j=1}^{n/2} (w^2)^{pj} \left(a_j + a_{j+\frac{n}{2}}\right).$$

Else: $i = 2p+1$.

$$b_i = \left(\sum_{j=1}^{n/2} w^{2 \cdot pj} \cdot a_j \cdot w^j\right) - \left(\sum_{j'=1}^{n/2} w^{2pj'} \cdot a_{j'+\frac{n}{2}} \cdot w^{j'}\right)$$

$$= \sum_{j=1}^{n/2} (w^2)^{pj} \cdot w^j \cdot \left(a_j - a_{j+\frac{n}{2}}\right)$$

$i \leftarrow$ odd

$$v_e = \begin{bmatrix} a_1 + a_{\frac{n}{2}+1} \\ a_2 + a_{\frac{n}{2}+2} \\ \vdots \\ a_{\frac{n}{2}} + a_n \end{bmatrix} \qquad v_o = \begin{bmatrix} w\left(a_1 - a_{\frac{n}{2}+1}\right) \\ w^2\left(a_2 - a_{\frac{n}{2}+2}\right) \\ \vdots \\ w^{n/2}\left(a_{\frac{n}{2}} - a_n\right) \end{bmatrix}$$

$DFT_n(\bar{a})$ from $DFT_{\frac{n}{2}}(v_e)$ and $DFT_{\frac{n}{2}}(v_o)$.

$$\left.\begin{bmatrix} b_1 \\ \\ b_n \end{bmatrix}\right\} \text{even terms} \quad \begin{bmatrix} b_2 \\ b_4 \\ \vdots \\ b_n \end{bmatrix} = DFT_{\frac{n}{2}}(v_e).$$

$$\begin{bmatrix} b_1 \\ b_3 \\ \vdots \\ b_{n-1} \end{bmatrix} = DFT_{\frac{n}{2}}(v_o)$$

$FFT_n(\bar{a})$: Fast (Discrete) Fourier Transform

W.L.O.G n is a power of 2.

If n is not a power of 2, append 0's at the bottom to make the vector 2^k

- Find primitive n^{th} root of unity
- Construct vectors v_e and v_o.
- Recursively call $FFT_{\frac{n}{2}}(v_e)$ and $FFT_{\frac{n}{2}}(v_o)$
- "Put together" the results
- Return $\bar{b}$ thus obtained.

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n) \longrightarrow O(n\log n).$$

$$P(z) = \sum_{i=0}^{n} c_i z^i \qquad Q(z) = \sum_{i=0}^{n} c_i' z^i \qquad \Big| \qquad R(z) = \sum_{i=0}^{2n} e_i z^i$$

$$\downarrow \qquad\qquad\qquad \downarrow$$

$$P(z) = \sum_{i=0}^{2n} c_i \cdot z^i \qquad Q(z) = \sum_{i=0}^{2n} c_i' \cdot z^i$$

s.t $c_i = 0 \ \forall i > n$ s.t $c_i' = 0 \ \forall i > n$

$w \leftarrow 2n^{th}$ primitive root

$$\left.\begin{matrix} P(w), \dots, P(w^{2n}) \\ Q(w), \dots, Q(w^{2n}) \end{matrix}\right\} R(w) \dots R(w^{2n}) \xrightarrow{\text{Inv DFT}} \text{coefs of } R.$$

$$2 \cdot T(2n) + T_{inv}(2n)$$

[De, Kurur, Saha, Saptharishi STOC 2008]

[Fürer, ~]

Best integer mult. algos