Discrete Fourier Transform.

$$b_i = \sum_{j=0}^{n-1} \omega^{ij} \cdot a_j.$$

- Ring $(R, +, \times, a\text{-id}, m\text{-id})$.

R

$$\vec{a} \xrightarrow{DFT} \vec{b} = DFT(\vec{a})$$
$$\phi \downarrow \qquad \phi \downarrow$$
$$\phi(\vec{a}) = \vec{a}' \xleftarrow{\text{inv DFT}} \phi(\vec{b})$$

Input: $\vec{a} = (a_0, a_1, \dots, a_{n-1})$

Discrete Fourier Transform $\left(DFT(\vec{a})\right) = A \cdot (\vec{a})^T$.

$\forall \; i,j \in [0, n-1], \quad A_{i,j} = \omega^{i \cdot j}$.

Assume that $\frac{1}{n}$ is available in our ring R. Assume that n is a power of 2.

ω: Primitive n^{th} root of unity

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

$$(A^{-1})_{i,j} = \frac{\omega^{-i \cdot j}}{n}.$$

Want: To compute the linear transform $(DFT(\vec{a}))$.

Remark: Naively, $\vec{b}$ needs $O(n^2)$ operations.

$DFT(\text{inv DFT}(\vec{a})) = \vec{a}$

$\text{inv DFT}(DFT(\vec{a})) = \vec{a}$

Rather any linear transformation takes at most $O(n^2)$ operations

Cooley-Tukey: If we consider a linear transformation by the DFT matrix, it can be done in $O(n \log n)$ operations.

$$(b_i) = \sum_{j=0}^{n-1} \omega^{ij} \cdot a_j$$

$$P_a(x) = \sum_{j=0}^{n-1} a_j x^j.$$

$$A \cdot (\vec{a})^T$$

$$b_0 = \sum_{i=0}^{n-1} (\omega^0)^j \cdot a_j = P_a(\omega^0)$$

$$\begin{bmatrix} b_0 \\ \\ b_{n-1} \end{bmatrix} = \begin{bmatrix} \omega^0 & \omega^0 & & \omega^0 \\ \omega^0 & \omega^1 & \cdots & \omega^{n-1} \\ \omega^0 & \omega^2 & & \omega^{2(n-1)} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \\ a_{n-1} \end{bmatrix}$$

↑ Evaluation of $P_a(x)$ at points $\omega^0, \omega^1, \ldots, \omega^{n-1}$.

$$\begin{bmatrix} P_a(\omega^0) \\ \\ P_a(\omega^{n-1}) \end{bmatrix} = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} \begin{bmatrix} a_0 \\ \\ a_{n-1} \end{bmatrix}$$

$n' = 2^k$

$$P_a'(x) = \sum_{j=0}^{n'-1} a_j x^j \quad \text{s.t } a_j = 0 \ \forall j > n-1.$$

$(a_0, \ldots, a_{n-1})$

$(a_0, \ldots, a_{n-1}, 0, \ldots)$

↳ A which is of size $n' \times n'$.

Given two polynomials $P(x)$ and $Q(x)$ (degree at most $n-1$): Compute their product

$$P(x) = A + x^{n/2} B$$
$$Q(x) = A' + x^{n/2} B'$$

$$PQ(x) = AA' + BB' \cdot x^n + (AB' + BA') x^{n/2}$$

↳ $n^{\log_2 3} \sim n^{1.\sim}$

$P(x)$ —DFT→ evaluations of P at $\{\omega^i\}_{i \in [0, 2n-2]}$

$Q(x)$ —DFT→ evals of Q at $\{\omega^i\}_{i \in [0, 2n-2]}$

Evaluations of $P(x) \cdot Q(x)$ at $\{\omega^i\}$

↓ InvDFT

Obtain coef. of the product.

$w^i \longrightarrow P(w^i) \cdot Q(w^i)$

$R(x) = P(x) \cdot Q(x)$

$= \left(\sum_{i=0}^{n-1} p_i \cdot x^i\right) \cdot \left(\sum_{j=0}^{n-1} q_j x^j\right)$

$\leq 2n-2$

Suff: Consider $2n^{th}$ root of unity.

$P = (p_0, \ldots, p_{n-1}, \underbrace{0, \ldots, 0}_{2n})$

$Q = (q_0, \ldots, q_{n-1}, \underbrace{0, \ldots, 0}_{2n})$

$\xrightarrow{DFT}$ $P(w^0), \ldots, P(w^{2n-1})$ and $Q(w^0), \ldots, Q(w^{2n-1})$

$R(w^0), \ldots, R(w^{2n-1})$

↓ Recover

Polynomial R.

$P \cdot Q \xrightarrow{DFT}$ evals(P), eval(Q) $\xrightarrow{O(n)}$ evals(R) $\xrightarrow{invDFT}$ R

$O(n \log n)$ $O(n \log n)$ $O(n^2)$

$O(n \log n)$ algorithm

$$b_i = \sum_{j=0}^{n-1} w^{ij} \cdot a_j$$

$$= \sum_{j=0}^{n/2-1} w^{ij} a_j + \sum_{j=n/2}^{n-1} w^{ij} \cdot a_j \qquad j = \frac{n}{2} + j' \quad n-1 = \frac{n}{2} + j'$$

$$= \sum_{j=0}^{n/2-1} w^{ij} a_j + w^{\frac{n}{2} i} \cdot \sum_{j'=0}^{\frac{n}{2}-1} w^{ij'} \cdot a_{j'+\frac{n}{2}}$$

$w^k \neq 1 \ \forall \ k \in [0, n-1]$. $w^n = 1$.

$\sqrt{w^n}$

Obs: w^2 is $\frac{n^{th}}{2}$ prim root of unity if w is n^{th} prim root of unity.

Obs: $w^{n/2} = (-1)$

$w^n = 1$.

$$= \sum_{j=0}^{\frac{n}{2}-1} \omega^{ij} \cdot a_j + (-1)^i \cdot \sum_{j'=0}^{\frac{n}{2}-1} \omega^{ij'} \cdot a_{j'+\frac{n}{2}}$$

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

$$b_i = \sum_{j=0}^{\frac{n}{2}-1} (\omega^{2p})^{j} a_j + \sum_{j'=0}^{\frac{n}{2}-1} (\omega^{2p})^{j'} \cdot a_{j'+\frac{n}{2}}.$$

$$= \sum_{j=0}^{\frac{n}{2}-1} (\omega^2)^{p \cdot j} \left(a_j + a_{j+\frac{n}{2}} \right).$$

$$= \sum_{j=0}^{\frac{n}{2}-1} (\omega^2)^{p \cdot j} \cdot C_j$$

$C_j = a_j + a_{j+\frac{n}{2}}$.

$\vec{C} = (C_0, \ldots, C_{\frac{n}{2}-1})$.

If i is odd

$$b_i = \sum_{j=0}^{n/2-1} \omega^{(2p+1)j} \cdot a_j - \sum_{j'=0}^{\frac{n}{2}-1} \omega^{(2p+1)j'} \cdot a_{j'+\frac{n}{2}}$$

$$= \sum_{j=0}^{\frac{n}{2}-1} (\omega^{2p+1})^{j} \left(a_j - a_{j+\frac{n}{2}} \right)$$

$A^{-1}_{ij} = \frac{\omega^{-ij}}{n}$

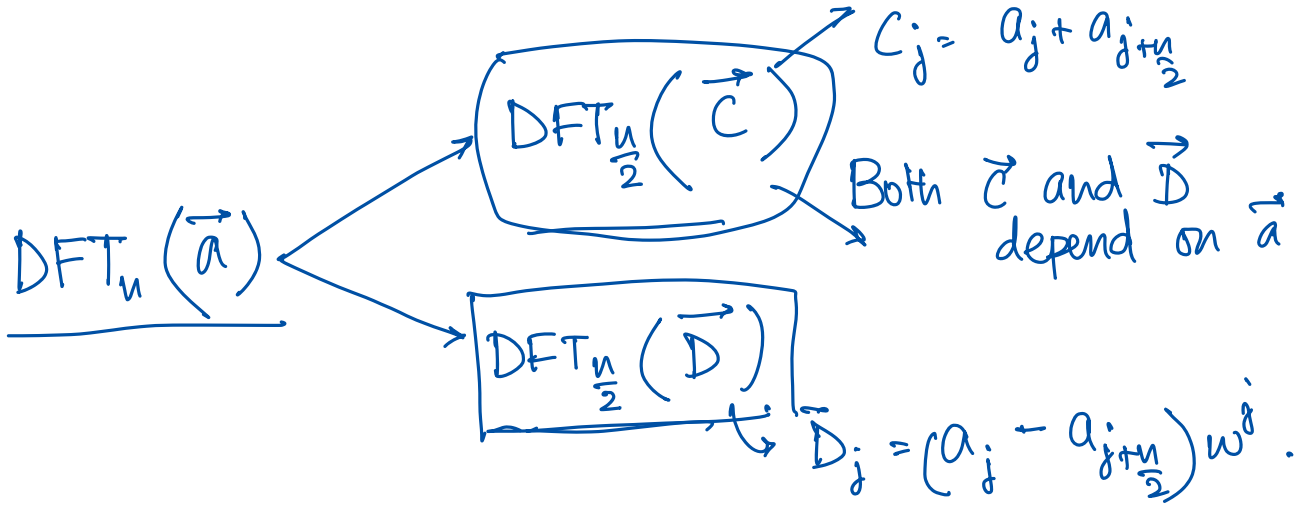
$$= \sum_{j=0}^{n/2-1} (\omega^2)^{p \cdot j} \left(\omega^j \right) \cdot \left(a_j - a_{j+\frac{n}{2}} \right)$$

$$= \sum_{j=0}^{n/2-1} (\omega^2)^{p \cdot j} D_j$$

where $D_j = \omega^j \cdot \left(a_j - a_{j+\frac{n}{2}} \right)$

$\vec{D} = (D_0, \ldots, D_{\frac{n}{2}-1})$

Even locations of $\vec{b}$ can be obtained by $DFT_{\frac{n}{2}}(\vec{C})$ and Odd locations from $DFT_{\frac{n}{2}}(\vec{D})$.



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

$$= O(n \log n).$$

Rewriting Polynomial mult:

Convolution of two vectors $\vec{a}$ and $\vec{b}$ (denoted by $\vec{a} * \vec{b}$)

$$i \in [0, 2n-2]; \qquad (a*b)_i = \sum_{\substack{j+k=i \\ 0 \le j,k \le n-1}} a_j \cdot b_k$$

Convolution gives coefs of the product of the polys whose coefs are $\vec{a}$ and $\vec{b}$.

$$\left(\sum_{j=0}^{n-1} a_j x^j\right)\left(\sum_{k=0}^{n-1} b_k x^k\right) = \sum_i \left(\sum_{j+k=i} a_j b_k\right) \cdot x^i$$

Convolution

$$a * b = \text{InvDFT}_{2n}\left(\text{DFT}_{2n}(\vec{a}) \odot \text{DFT}_{2n}(\vec{b})\right)$$

$\odot$ ← point wise mult.

$(a_1, \ldots, a_n) \odot (b_1, \ldots, b_n) = (a_1 b_1, a_2 b_2, \ldots, a_n b_n)$

$\vec{a}' = (a_0, \ldots, a_{n-1}, \underbrace{0, \ldots, 0}_{2n})$

$\vec{b}' = (b_0, \ldots, b_{n-1}, \underbrace{0, \ldots, 0}_{2n})$.

Using DFT: Integer mult is in $O(n \log n)$ if $\mathcal{R}$ is "nice".

↓ Else

$O(n \log n \log\log n)$ ← Schonhage-Strassen integer mult.

↓

$O(n \log n) \cdot 2^{O(\log^* n)}$ → [Furer 2008]

{[De-Kurur-Saha-Saptarishi] STOC 2008