15-750
3/3/10

Fast Fourier Transform (FFT) & Polynomials

FFT: 1) Critical in signal processing

2) Used in latest string matching Algorithms

3) Image Compression

EG. Use FFT to quickly Mult two Polynomials

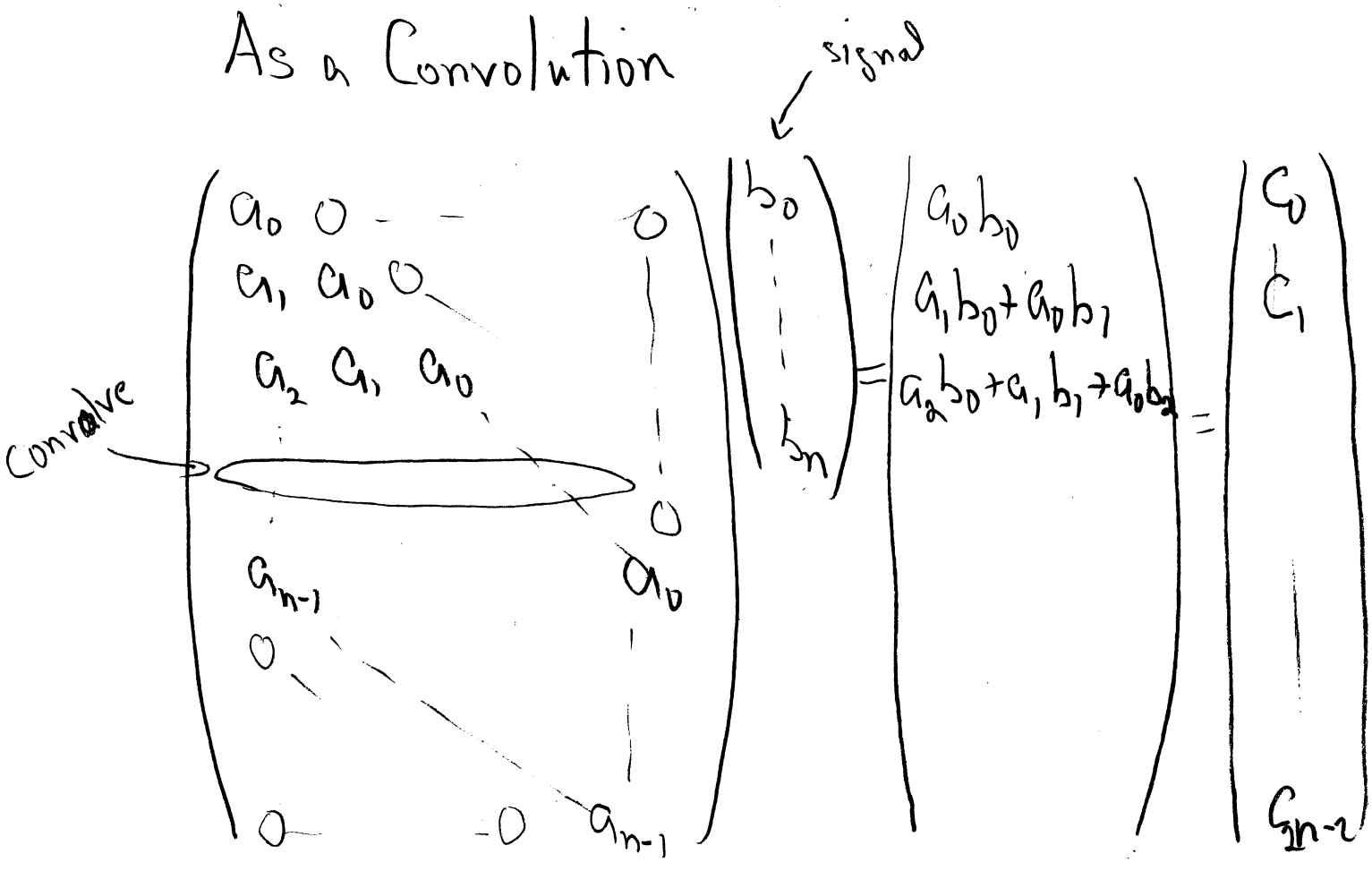
$$f(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$$
$$g(x) = b_0 + b_1 x + \cdots + b_{n-1} x^{n-1}$$

Let $h(x) = f(x) \cdot g(x)$

$$h(x) = c_0 + c_1 x + \cdots - c_{2n-2} x^{2n-2}$$

$$c_k = \sum_{i+j=k} a_i b_j \qquad O(n^2) \text{ Arith Ops.}$$

As a Convolution



convolve

signal b_0 b_1 b_{n-1} - - a_0 a_1 b

Gaussian Smoothing

Making the Impossible Work!

Idea:

1) Evaluate: $f(0), f(1), \cdots, f(2m-2), g(0), \cdots, g(2m-2)$

2) Mult: $r_0 = f(0)g(0), \cdots, r_{2m-2} = f(2m-2)\,g(2m-2)$

3) Interpolate:

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 2^2 & \cdots & \\ \vdots & & & & \\ 1 & 2m-2 & (2m-2)^2 & \cdots & \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{2m-2} \end{pmatrix} = \begin{pmatrix} r_0 \\ \vdots \\ r_{2m-2} \end{pmatrix}$$

is solve ↑

Steps 1) & 3) look at best $O(n^2)$ $n = 2m-1$

Only 2) is $O(n)$

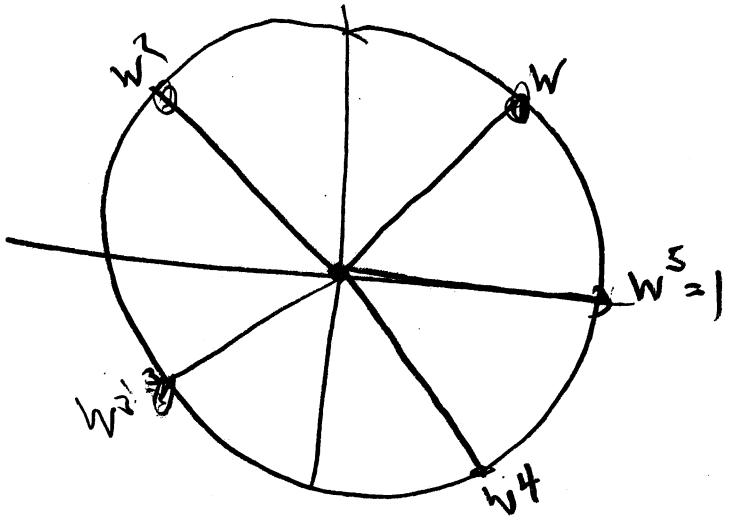
Idea: Pick nth-roots of unity to evaluate polynomials

The n-th roots of unity

Complex plane $a+ib$ $a,b \in \mathbb{R}$ $i^2=-1$

$a=\cos\left(\frac{2\pi}{n}\right)$ $b=\sin\left(\frac{2\pi}{n}\right)$ $w=a+ib$

eg $n=5$



Claim: $w=\cos\theta+i\sin\theta$ then

$$w^2=\cos 2\theta+i\sin 2\theta$$

pf Let R_θ = CCW rotation of plane by θ degrees.

$$R_\theta=\begin{pmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{pmatrix}=M \qquad R_{2\theta}=M^2=\begin{pmatrix}\cos^2\theta-\sin^2\theta & *\\ 2\cos\theta\sin\theta & *\end{pmatrix}$$

n-th roots of unity Definition

W is an n-th root if

1) $W^p \neq 1$ for $1 \leq p < n$

2) $W^n = 1$

Claims

3) $\sum_{j=0}^{n-1} W^j = 0$ pf $X = \sum_{j=0}^{n-1} W^j$

$$WX = \sum_{j=0}^{n-1} W^j = X \Rightarrow X = 0$$

---

4) $\sum_{j=0}^{n-1} W^{jp} = 0$ for $1 \leq p < n$

pf set $W = w^p$

and repeat proof in 3)

Def DFT

numbering $0, \dots, n-1$

$$F_n = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{n-1} \\ 1 & w^2 & w^4 & \cdots & w^{2(n-1)} \\ 1 & w^3 & w^6 & \cdots & \\ \vdots & & & & \\ 1 & w^{n-1} & \cdots & & w^{(n-1)^2} \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} v_0 \\ \vdots \\ v_{n-1} \end{pmatrix}$$

i.e. $(F_n)_{ij} = w^{ij} \quad 0 \le i, j \le n-1$ $\quad n=2$, $F_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

$n=4$

$$F_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix} \begin{pmatrix} a_{00} \\ a_{01} \\ a_{10} \\ a_{11} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & i & -i \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -i & i \end{pmatrix} \begin{pmatrix} a_{00} \\ a_{10} \\ a_{01} \\ a_{11} \end{pmatrix} \begin{matrix} \} \text{ even} \\ \\ \} \text{ odd} \end{matrix}$$

$$= \begin{pmatrix} F_2 & C \\ F_2 & -C \end{pmatrix} \begin{pmatrix} \quad \\ \quad \end{pmatrix} = \begin{pmatrix} F_2 & \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} F_2 \\ F_2 & -\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} F_2 \end{pmatrix}$$

Thm if we reorder columns even, odd F_n is

$$\left(\begin{array}{c|c} F_{n/2} & D_{n/2} F_{n/2} \\ \hline F_{n/2} & -D_{n/2} F_{n/2} \end{array}\right) \qquad D_{n/2} = \begin{pmatrix} 1 & & & 0 \\ & w & & \\ & & w^2 & \\ 0 & & & \ddots \\ & & & & w^{n/2-1} \end{pmatrix}$$

(even) (odd)

---

no proof

$$F_{2n}(a) = \begin{pmatrix} F_n & D_n F_n \\ F_n & -D_n F_n \end{pmatrix} \begin{pmatrix} a_{even} \\ a_{odd} \end{pmatrix}$$

$$= \begin{pmatrix} F_n a_{even} + D_n F_n a_{odd} \\ F_n a_{even} - D_n F_n a_{odd} \end{pmatrix}$$

Recursive form of FFT

$T(n) = 2T(n/2) + cn$

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

Back to Poly Multi

We will use n-th roots of unity

step 1 $O(n \log n)$ time

step 2 $O(n)$ time

step 3 ie solve $F_n a = b$ for a

Trick: F_n^{-1} is just an FFT

Claim $\left(F_n^{-1}\right)_{ij} = \left(\frac{1}{n}\right) w^{-ij} = \frac{1}{n} F_n(w^{-1})$

$= \left(\frac{1}{n}\right)(w^{-1})^{ij}$

FFT inverse

Thm $\left(F_n^{-1}\right)_{ij} = \frac{1}{n} W^{-ij}$

if $\sum_{j=0}^{n-1} W^{ij} W^{-jk} = \begin{cases} n & \text{if } i = k \\ 0 & 0.w \end{cases}$

$$LHS = \sum_{j=0}^{n-1} W^{j(i-k)}$$

if $i = k$ then $LHS = \sum_{j=0}^{n-1} 1 = n$

if $i \neq k$ then $LHS = \sum_{j=0}^{n-1} W^{jp} = 0$ prop 4)

$1 \leq p < n$

Thus step 3) is just FFT using different root (w^{-1})

Thm Two deg n polys can be multi in $O(n \log n)$ ops.

2-issues

A) Do we have to work with complex number?

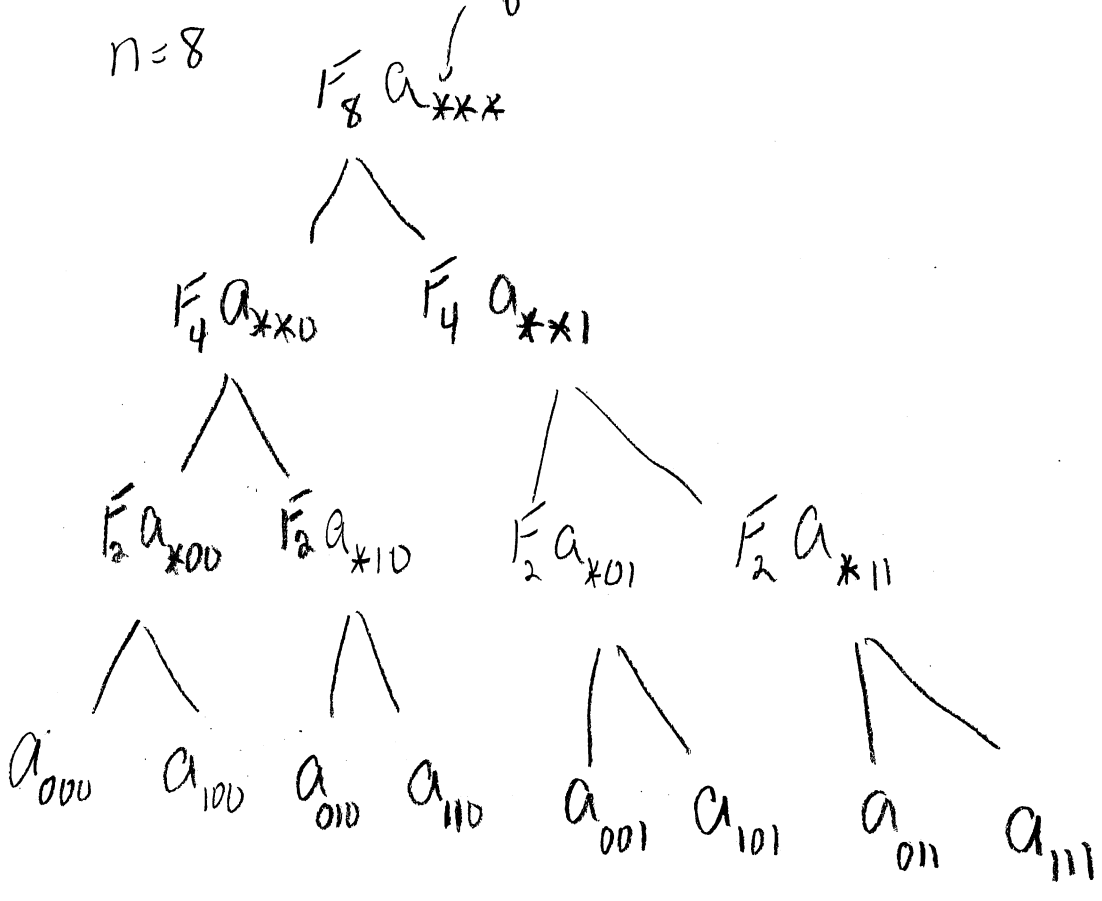
B) How do we arrange the FFT alg.

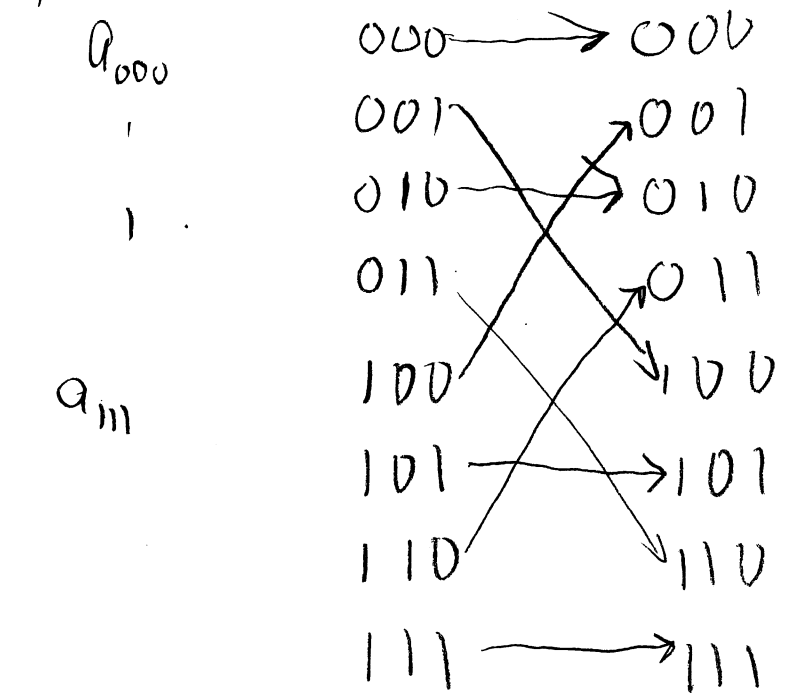
A) Quotient Rings

B) FFT in-Place

$n = 8$

high order bit.





Bit Reverse Permutation

After Bit Reverse FFT can be done in place