

COL 351:

Analysis and Design of Algorithms

Lecture 27

Polynomial Multiplication

Given: Two polynomials $A(x) = a_0 + a_1x + \dots + a_nx^n$ and $B(x) = b_0 + b_1x + \dots + b_nx^n$, with degree less than equal to 'n' and integer coefficients.

Find: Product $A(x) \cdot B(x) = c_0 + c_1x + c_2x^2 + \dots + c_{2n}x^{2n}$ (Say, $C(x)$)

Definition: Convolution

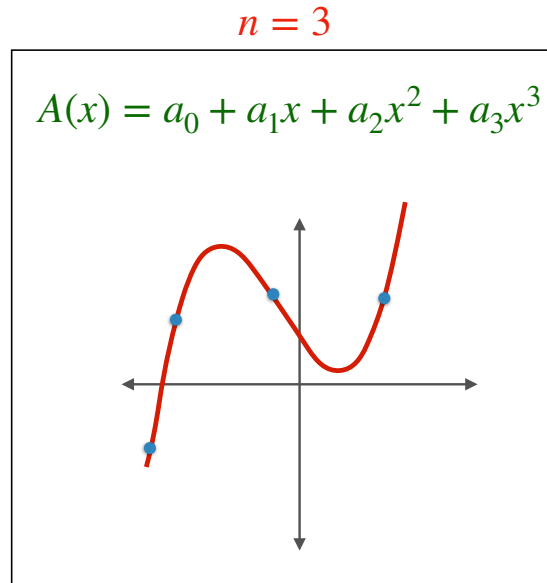
Vector $[c_0, c_1, c_2, \dots, c_{2n}]$ is referred as
“Convolution” of vectors
 $[a_0, a_1, \dots, a_n]$ and $[b_0, b_1, \dots, b_n]$.

$$c_i = (a_0 b_i) + (a_1 b_{i-1}) + (a_2 b_{i-2}) + \dots + (a_i b_0) = \sum_{j=0}^i a_j b_{i-j}$$

Trivial: $O(n^2)$

Representation of a polynomial

- An alternate way to represent polynomial $A(x) = a_0 + a_1x + \cdots + a_nx^n$.



Evaluation at

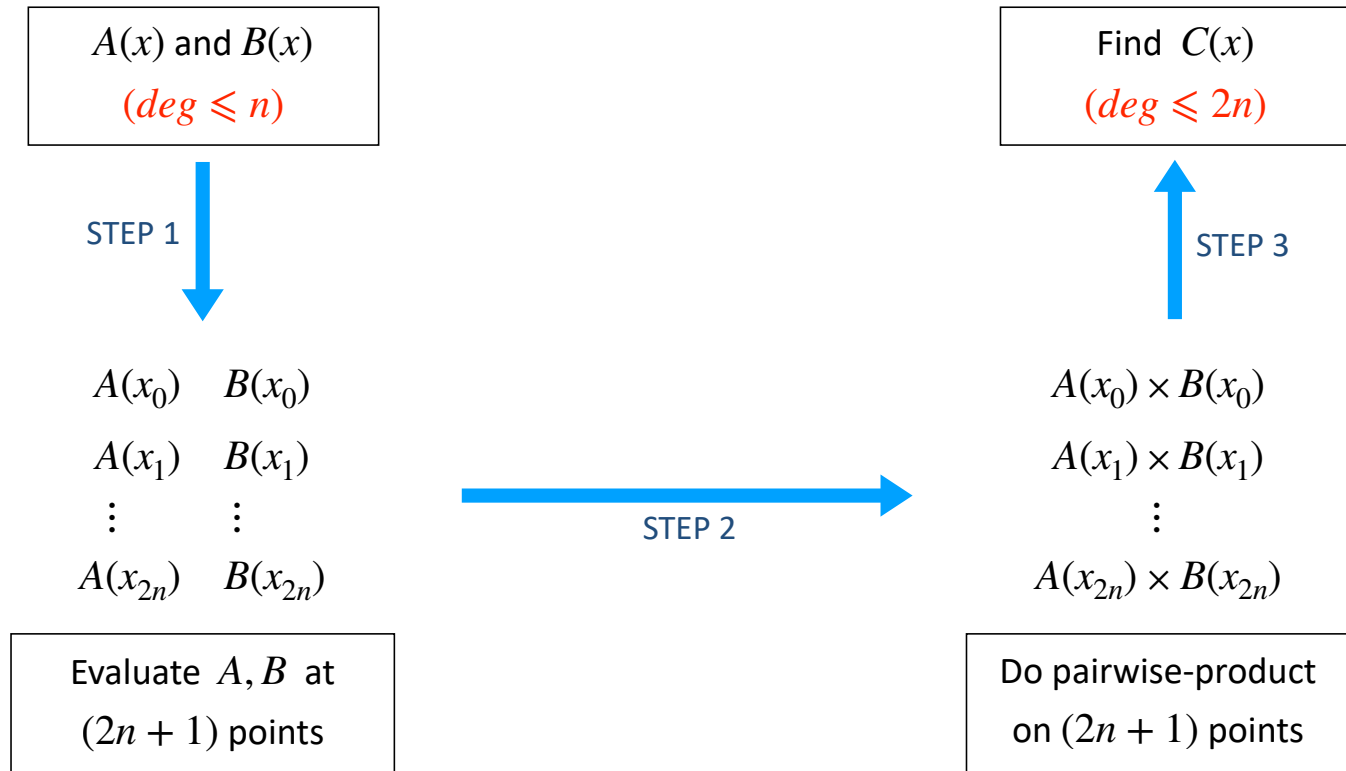
$$N = n + 1 = 4$$

point suffices

Why are we looking at alternate representation?

- Answer: Efficient way to compute product.

Take a set $S = \{x_0, x_1, x_2, \dots, x_{2n}\}$



Point-wise evaluation

$|S| \leq N$ and $\deg < N$

Aim: Given a polynomial 'A' of degree $\leq n$, find its evaluation on $S = \{1, \omega, \omega^2, \dots, \omega^{N-1}\}$.

$$\begin{aligned} A(x) &= a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 \dots + a_nx^n \\ &= (a_0 + a_2x^2 + a_4x^4 + \dots) + x(a_1 + a_3x^2 + a_5x^4 + \dots) \\ &= A_{\text{even}}(x^2) + x \cdot A_{\text{odd}}(x^2) \end{aligned}$$

Assume $N = (n + 1)$
is a power of 2

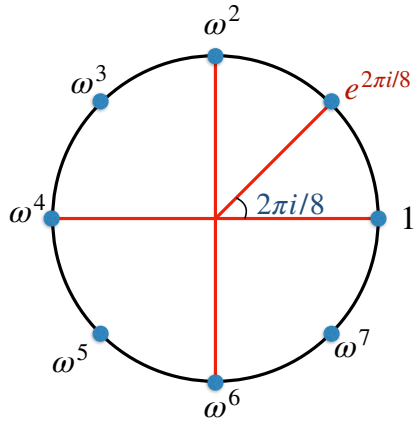
Remark 1: Degree of polynomials $A_{\text{even}}, A_{\text{odd}} \leq (n - 1)/2 < N/2$.

Remark 2: As $|S^2| = N/2$, we get two subproblems of size $N/2$.

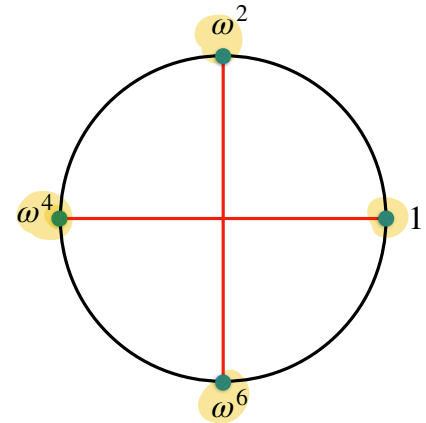
Example

$N = 8$

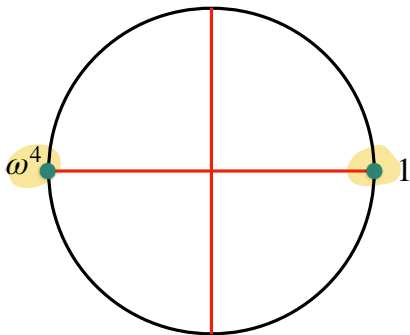
Set S is
roots of
 $x^8 = 1$



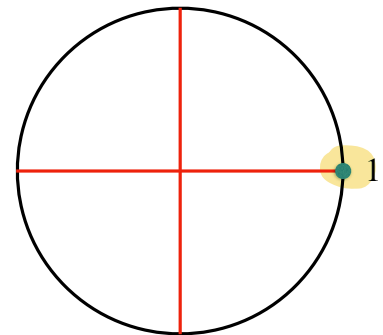
Set S^2



Set S^4



Set S^8



Discrete Fourier Transform (DFT)

Definition: Let $A(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$ be a polynomial of degree n .

Then $DFT([a_0, a_1, \dots, a_n]) = [A(1), A(\omega), A(\omega^2), \dots, A(\omega^n)]$

Fast Fourier Transform (FFT):

Divide and conquer algorithm to compute DFT

Inverse DFT

Definition: Given the evaluations y_0, y_1, \dots, y_n on $(n+1)^{th}$ roots of unity $\{1, \omega, \dots, \omega^n\}$, find the corresponding polynomial of degree n .

Then *Inverse – DFT* $([y_0, y_1, \dots, y_n]) = [a_0, a_1, \dots, a_n]$.

How to compute inverse DFT?

Definition: Primitive Root

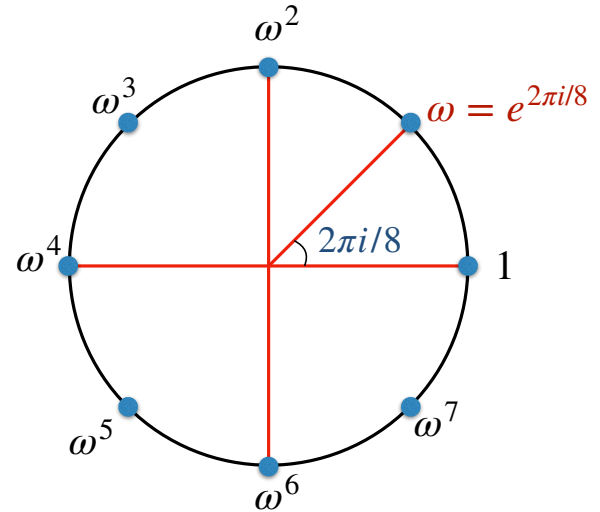
Primitive Root:

An N^{th} root of unity that can generate all other N^{th} roots.

N^{th} primitive root of unity:

ω such that

- $\omega^N = 1$, and
- $\omega^i \neq 1$, for $0 < i < N$



Properties of a primitive root

Claim 1:

If ω is N^{th} root of unity other than 1, then show that $1 + \omega + \dots + \omega^{N-1} = 0$.

$$= \frac{\omega^N - 1}{\omega - 1} = \frac{0}{\omega - 1} = 0.$$

Claim 2:

If ω is N^{th} primitive root of unity, then for $i \in [1, N-1]$ we have

$$1 + \omega^i + \dots + \omega^{i(N-1)} = 0.$$

Let $x = \omega^i$. Then x is N^{th} root other than 1.

By Claim 1, $1 + \omega^i + \dots + \omega^{i(N-1)} = 1 + x + \dots + x^{N-1} = 0$

Discrete Fourier transform

evaluations

$$\begin{bmatrix} A(1) \\ A(\omega^1) \\ A(\omega^2) \\ \vdots \\ A(\omega^i) \\ \vdots \\ A(\omega^n) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^j & \dots & \omega^n \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2j} & \dots & \omega^{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & \omega^i & (\omega^i)^2 & \dots & (\omega^i)^j & \dots & (\omega^i)^n \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & \omega^n & \omega^{2n} & \dots & \omega^{nj} & \dots & \omega^{nn} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_j \\ \vdots \\ a_n \end{bmatrix}$$

coefficients

DFT matrix w.r.t. ' ω '

ω is $N = (n + 1)^{th}$
primitive root of unity

Inverse Discrete Fourier transform

coefficients \rightarrow

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_j \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^j & \dots & \omega^n \\ 1 & & & & & & \\ \vdots & & & & & & \\ 1 & \omega^i & (\omega^i)^2 & \dots & (\omega^i)^j & \dots & (\omega^i)^n \\ \vdots & & & & & & \\ 1 & & & & & & \end{bmatrix}^{-1} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{bmatrix}$$

evaluations \leftarrow

\uparrow
Inverse of DFT matrix w.r.t. 'w'

ω is $N = (n + 1)^{th}$
primitive root of unity

*How to
compute
inverse matrix?*

Inverse of DFT Matrix

DFT - matrix (ω)

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^k & \dots & \omega^n \\ 1 & & & & & & \\ \vdots & & & & & & \\ 1 & \omega^i & (\omega^i)^2 & \dots & (\omega^i)^k & \dots & (\omega^i)^n \\ \vdots & & & & & & \\ 1 & & & & & & \end{bmatrix}$$

DFT - matrix (ω^{-1})

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 & \dots & 1 \\ 1 & \omega^{-1} & (\omega^{-1})^2 & \dots & (\omega^{-1})^j & \dots & (\omega^{-1})^n \\ 1 & & & & & & \\ \vdots & & & & & & \\ 1 & \omega^{-k} & (\omega^{-k})^2 & \dots & (\omega^{-k})^j & \dots & (\omega^{-k})^n \\ \vdots & & & & & & \\ 1 & & & & & & \end{bmatrix}$$

$$(i, j)^{th} \text{ Term} = \sum_{k=0}^n (\omega^i)^k (\omega^{-j})^k = \sum_{k=0}^n (\omega^{i-j})^k = \begin{cases} N & \text{if } i=j \\ 0 & \text{if } i \neq j \end{cases}$$

Inverse Discrete Fourier transform

DFT - matrix (ω^{-1})

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_j \\ \vdots \\ a_n \end{bmatrix} = \frac{1}{N} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 & \dots & 1 \\ 1 & \omega^{-1} & (\omega^{-1})^2 & \dots & (\omega^{-1})^j & \dots & (\omega^{-1})^n \\ 1 & \omega^{-2} & (\omega^{-2})^2 & \dots & (\omega^{-2})^j & \dots & (\omega^{-2})^n \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & \omega^{-i} & (\omega^{-i})^2 & \dots & (\omega^{-i})^j & \dots & (\omega^{-i})^n \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & \omega^{-n} & (\omega^{-n})^2 & \dots & (\omega^{-n})^j & \dots & (\omega^{-n})^n \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{bmatrix}$$

ω is $N = (n+1)^{th}$
primitive root of unity

↑
This can be multiplied
in $O(N \log N)$ time
using FFT algorithm
S will be $\{1, \omega^{-1}, \dots, \omega^{-n}\}$

Inverse Discrete Fourier transform

Steps to compute Inverse-DFT of vector $[y_0, y_1, \dots, y_n]$:

1. Consider the polynomial $Y(x) = \frac{y_0 + y_1x + y_2x^2 + \dots + y_nx^n}{N}$

2. Let $S_{inv} = \{1, \omega^{-1}, \omega^{-2}, \dots, \omega^{-n}\}$ *Roots of poly $x^N = 1$.*

3. Evaluate $Y(x)$ over S_{inv} .

ω is $N = (n + 1)^{th}$
primitive root of unity

Main Result

Theorem:

Two polynomials of degree n with integer coefficients can be multiplied in $O(n \log n)$ time.

Assumption: We can do operations on $N = (n + 1)$ -th roots of unity in $O(1)$ time.

operations



In practice there are
rounding errors if
coefficients are large

Homework - Write pseudocode to compute DFT and Inverse-DFT.

Applications

Question:

Let $A = \{a_1, a_2, \dots, a_n\}$ and $B = \{b_1, b_2, \dots, b_n\}$ be sets of positive integers in range $[1, M]$. Compute the set $S = \{x + y \mid x \in A, y \in B\}$ in $O(M \log M)$ time.