COL 351: Analysis and Design of Algorithms

Lecture 27

Polynomial Multiplication

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

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

Definition: Convolution

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

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

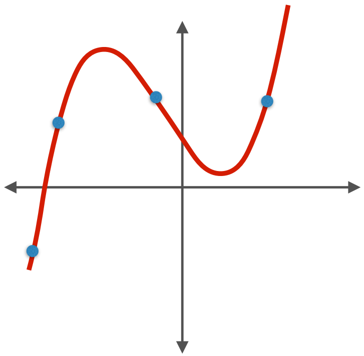
Trivial: $O(n^2)$

Representation of a polynomial

- An alternate way to represent polynomial $A(x) = a_0 + a_1 x + \cdots + a_n x^n$.

$n = 3$

$$A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$$



Evaluation at

$N = n + 1 = 4$

point suffices

Why are we looking at alternate representation?

- Answer: Efficient way to compute product.
  Take a set $S = \{x_0, x_1, x_2, \ldots, x_{2n}\}$

$A(x)$ and $B(x)$

$(deg \leqslant n)$

STEP 1

$$A(x_0) \quad B(x_0)$$
$$A(x_1) \quad B(x_1)$$
$$\vdots \qquad \vdots$$
$$A(x_{2n}) \quad B(x_{2n})$$

Evaluate A, B at $(2n+1)$ points

STEP 2

$$A(x_0) \times B(x_0)$$
$$A(x_1) \times B(x_1)$$
$$\vdots$$
$$A(x_{2n}) \times B(x_{2n})$$

Do pairwise-product on $(2n+1)$ points

STEP 3

Find $C(x)$

$(deg \leqslant 2n)$

Point-wise evaluation

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

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

$$
\begin{aligned}
A(x) &= a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 \cdots + a_n x^n \\
&= \left(a_0 + a_2 x^2 + a_4 x^4 + \cdots\right) + x\left(a_1 + a_3 x^2 + a_5 x^4 + \cdots\right) \\
&= A_{even}\left(x^2\right) + x \cdot A_{odd}\left(x^2\right)
\end{aligned}
$$

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

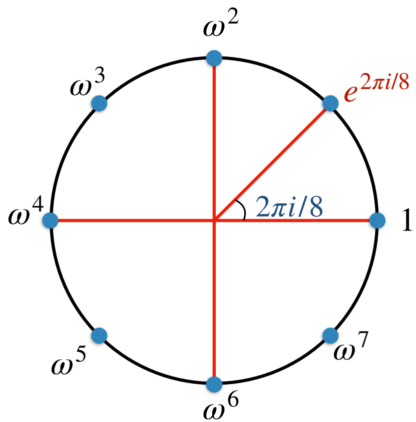
Remark 1: Degree of polynomials A_{even}, $A_{odd} \leqslant (n-1)/2 < N/2$.

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

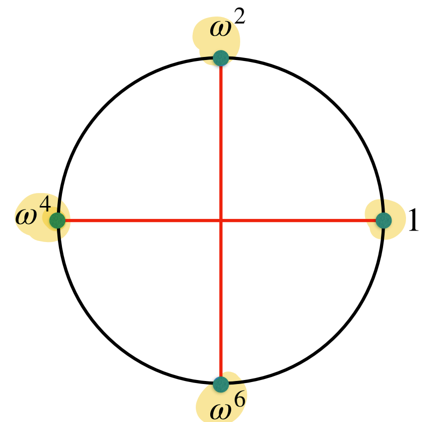
Example

N = 8

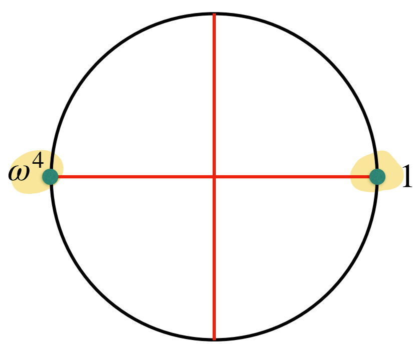
Set S is **roots of** $x^8 = 1$



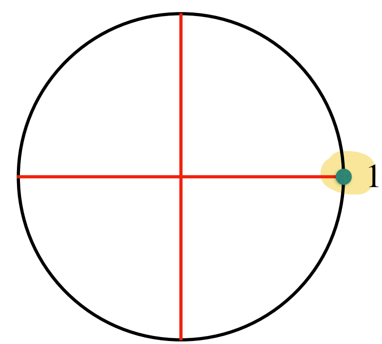
Set S^2



Set S^4



Set S^8



Discrete Fourier Transform (DFT)

Definition: Let $A(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$ be a polynomial of degree n.

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

Fast Fourier Transform (FFT):
Divide and conquer algorithm to compute DFT

Inverse DFT

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

Then $Inverse - DFT\big([y_0, y_1, \ldots, y_n]\big) = [a_0, a_1, \ldots, a_n]$.

How to compute inverse DFT?

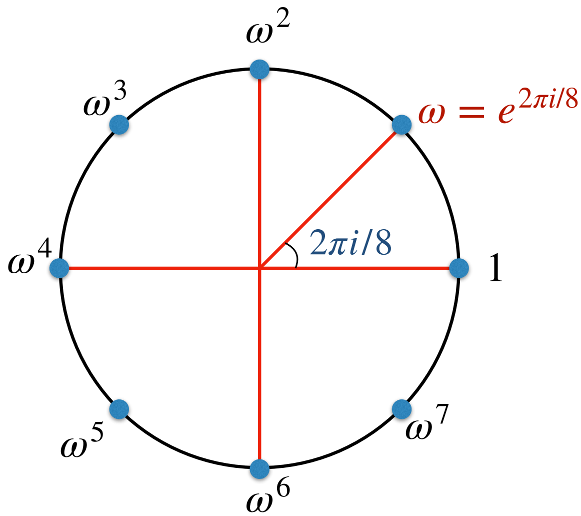
Definition: Primitive Root

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

Nth primitive root of unity:

ω such that

- $\omega^N = 1$, and
- $\omega^i \neq 1, \quad \text{for} \quad 0 < i < N$



Properties of a primitive root

Claim 1:

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

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

Claim 2:

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

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

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

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

Discrete Fourier transform

evaluations

coefficients

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

DFT matrix w.r.t. 'ω'

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

Inverse Discrete Fourier transform

coefficients

evaluations

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

Inverse of DFT matrix w.r.t. 'ω'

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

How to compute inverse matrix?

Inverse of DFT Matrix

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^k & \cdots & \omega^n \\ 1 & & & & & & \\ \vdots & & & & & & \\ 1 & \omega^i & (\omega^i)^2 & \cdots & (\omega^i)^k & \cdots & (\omega^i)^n \\ \vdots & & & & & & \\ 1 & & & & & & \end{bmatrix}}_{}^{\text{DFT - matrix } (\omega)} \overbrace{\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & \cdots & 1 \\ 1 & \omega^{-1} & (\omega^{-1})^2 & \cdots & (\omega^{-1})^j & \cdots & (\omega^{-1})^n \\ 1 & & & & & & \\ \vdots & & & & & & \\ 1 & \omega^{-k} & (\omega^{-k})^2 & \cdots & (\omega^{-k})^j & \cdots & (\omega^{-k})^n \\ \vdots & & & & & & \\ 1 & & & & & & \end{bmatrix}}^{\text{DFT - matrix } (\omega^{-1})}$$

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

Inverse Discrete Fourier transform

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

This can be multiplied in $O(N \log N)$ time using FFT algorithm

S will be $\{1, \omega^{-1}, \ldots, \omega^{-n}\}$

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

Inverse Discrete Fourier transform

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

1. Consider the polynomial $Y(x) = \dfrac{y_0 + y_1 x + y_2 x^2 + \cdots + y_n x^n}{N}$

2. Let $S_{inv} = \{1, \omega^{-1}, \omega^{-2}, \ldots, \omega^{-n}\}$ 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

operations

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.

In practice there are rounding errors if coefficients are large.

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

Applications

Question:

Let $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$ 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.