Algorithms and Data Structures

Fast Fourier Transform

Multiplying two polynomials

Suppose that we have two polynomials of degree n

$$A(x) = a_0 + a_1 x + a_2 x^2 + \ldots, + a_{n-1} x^{n-1}$$

$$B(x) = b_0 + b_1 x + b_2 x^2 + \ldots, + b_{n-1} x^{n-1}$$

The product is a polynomial $C(x)$ of degree $2n - 2$ where the coefficient of the term x^k is

$$c_k = \sum_{(i,j):i+j=k} a_i b_j$$

Example

Suppose that we have two polynomials of degree n

$$A(x) = a_0 + a_1x + a_2x^2 + \ldots, + a_{n-1}x^{n-1}$$

$$B(x) = b_0 + b_1x + b_2x^2 + \ldots, + b_{n-1}x^{n-1}$$

Coefficient of $x^0 = 1$ is a_0b_0

Coefficient of $x^1 = x$ is $a_0b_1 + a_1b_0$

Coefficient of x^2 is $a_0b_2 + a_1b_1 + a_2b_0$

Multiplying two polynomials

Suppose that we have two polynomials of degree n

$$A(x) = a_0 + a_1 x + a_2 x^2 + \ldots, + a_{n-1} x^{n-1}$$

$$B(x) = b_0 + b_1 x + b_2 x^2 + \ldots, + b_{n-1} x^{n-1}$$

The product is a polynomial $C(x)$ of degree $2n - 2$ where the coefficient of the term x^k is

$$c_k = \sum_{(i,j): i+j=k} a_i b_j$$

Equivalently: the coefficient vector c of $C(x)$ is the *convolution* $a * b$ of the coefficient vectors of $A(x)$ and $B(x)$.

How to compute $C(x)$?

Naive approach: Compute all the partial products (for every pair (i, j)) and add them up.

What is the running time in this case?

$\Theta(n^2)$

We will attempt to design a faster algorithm using Divide & Conquer.

Fast Fourier Transform (FFT)

Key idea: How to represent polynomials

Representation 1: via their coefficient vectors

$$a = (a_0, a_1, \ldots, a_{n-1}) \,, b = (b_0, b_1, \ldots, b_{n-1})$$

A different representation

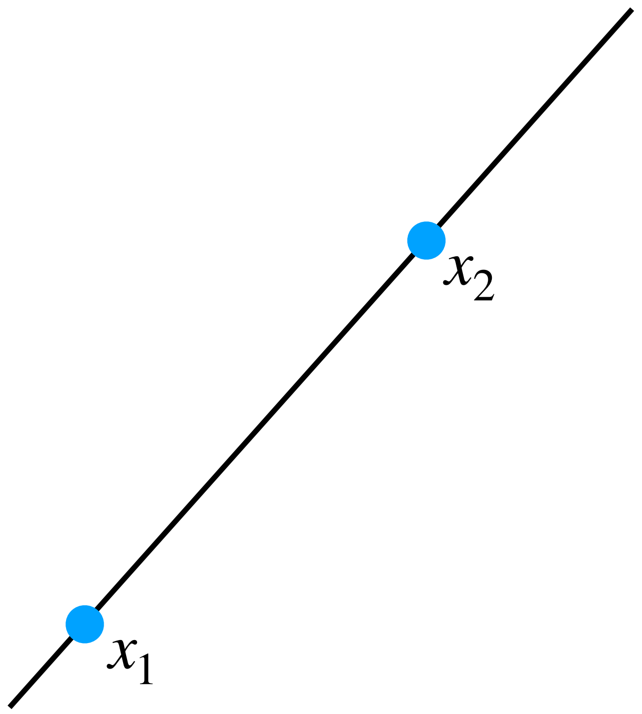
Consider the polynomial

$A(x) = a_0 + a_1 x$

What is this, geometrically?

What is the a way to represent a line uniquely?

Via two values x_1 and x_2.



Polynomial interpolation

Consider the polynomial
$A(x) = a_0 + a_1 x + a_2 x^2 + \ldots, a_d x^d$

Fact: Any polynomial of degree d can be represented by its values on at least $d + 1$ points.

Key idea: How to represent polynomials

Representation 1: via their coefficient vectors

$$a = (a_0, a_1, \ldots, a_{n-1}) \, , b = (b_0, b_1, \ldots, b_{n-1})$$

Representation 2: via their values on at least n points

New strategy

Step 1: Choose $2n$ values $x_1, x_2, \ldots, x_{2n}$ and *evaluate* $A(x_j)$ and $B(x_j)$ for each $j = 1,2,\ldots,2n$.

Step 2: Compute $C(x_j) = A(x_j) \cdot B(x_j)$ for all j *(these are now just numbers).*

Step 3: Recover C from $C(x_1), C(x_2), \ldots, C(x_{2n})$.

Running time

$\Omega(n)$ for each j

$\Omega(n^2)$ overall

Step 1: Choose $2n$ values $x_1, x_2, \ldots, x_{2n}$ and *evaluate* $A(x_j)$ and $B(x_j)$ for each $j = 1,2,\ldots,2n$.

Step 2: Compute $C(x_j) = A(x_j) \cdot B(x_j)$ for all j $O(n)$

(these are now just numbers).

Step 3: Recover C from $C(x_1), C(x_2), \ldots, C(x_{2n})$.

? no idea for now

A different representation

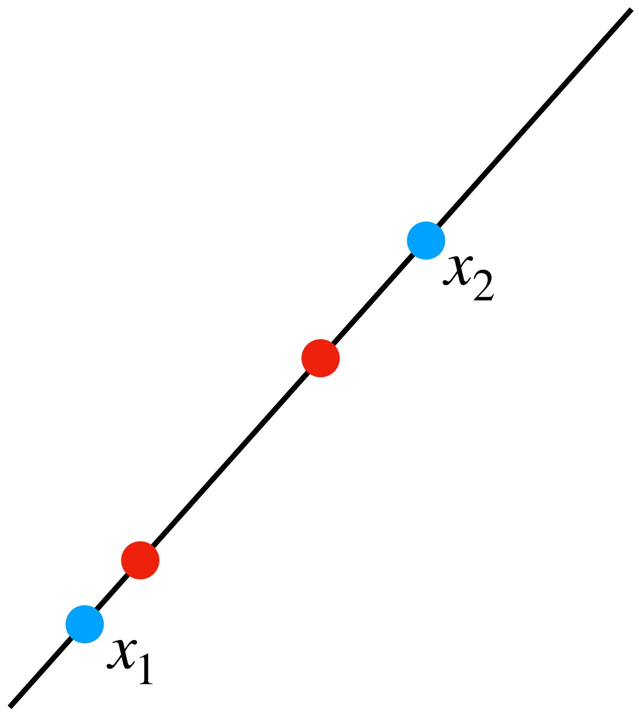
Consider the polynomial

$A(x) = a_0 + a_1 x$

What is this, geometrically?

What is the a way to represent a line uniquely?

Via two values x_1 and x_2.



Running time

$\Omega(n)$ for each j

$\Omega(n^2)$ overall

Step 1: Choose $2n$ values $x_1, x_2, \ldots, x_{2n}$ and *evaluate* $A(x_j)$ and $B(x_j)$ for each $j = 1,2,\ldots,2n$.

Step 2: Compute $C(x_j) = A(x_j) \cdot B(x_j)$ for all j $O(n)$
(these are now just numbers).

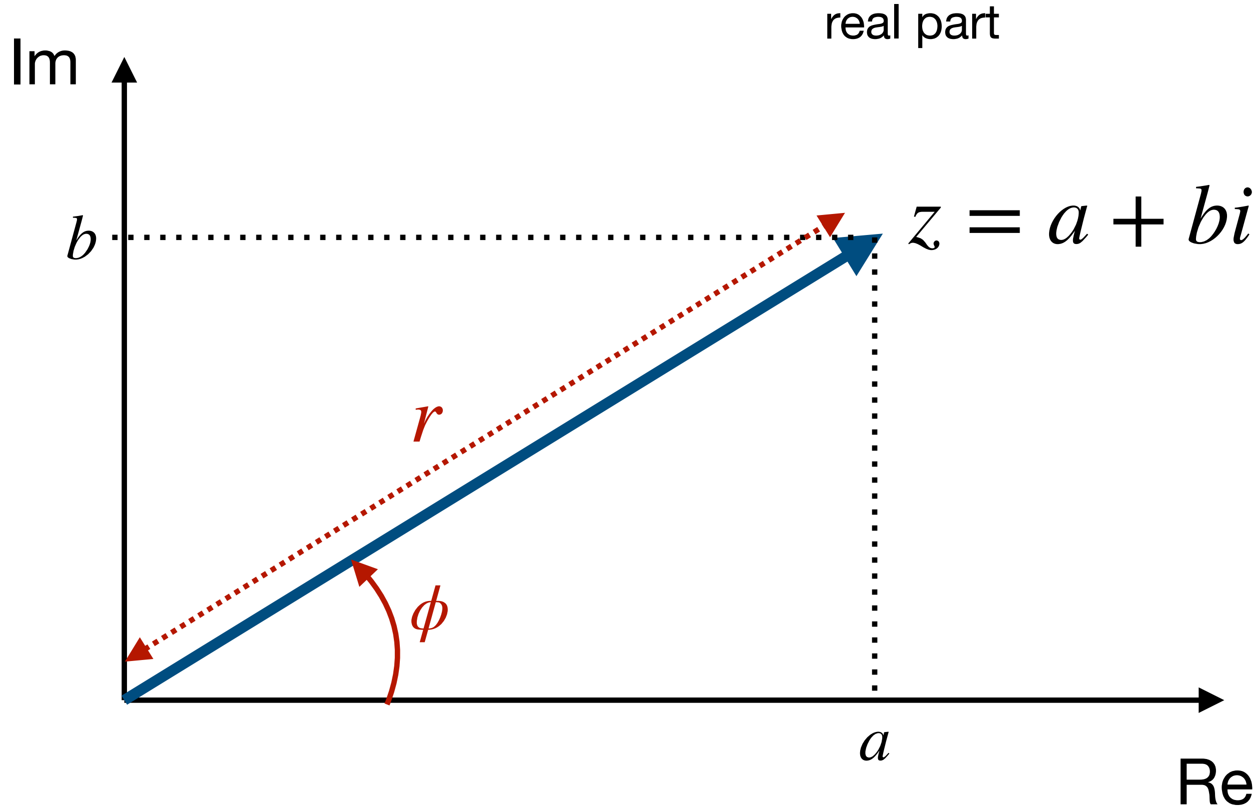
Step 3: Recover C from $C(x_1), C(x_2), \ldots, C(x_{2n})$.

? no idea for now

We will choose the $2n$ values carefully!

Quick Detour: Complex Numbers $\mathbb{C}$

Complex number $z = \underbrace{a}_{\text{real part}} + \overbrace{bi}^{\text{imaginary part}}$ $\quad i^2 = -1$



Polar Coordinates

$z = r(\cos\phi + i\sin\phi)$

Euler's formula: $e^{ix} = \cos x + i \sin x$

$z = r \cdot e^{i\phi}$

Argument ϕ : the angle of the radius with the positive real axis

Magnitude r : $r = |z| = \sqrt{a^2 + b^2}$

Roots of Unity

Let n be a positive integer. An nth root of unity is a (complex) number x satisfying the equation $x^n = 1$.

The nth roots of unity are:

$$\cos\frac{2k\pi}{n} + i\sin\frac{2k\pi}{n},\ \text{for } k = 0,1,\ldots,n-1$$

Equivalently: $e^{\frac{2k\pi i}{n}}$, for $k = 0,1,\ldots,n-1$

The quantity $e^{\frac{2\pi i}{n}} = \cos(2\pi/n) + i\sin(2\pi/n)$ is called the *principal nth root of unity.*

Roots of Unity

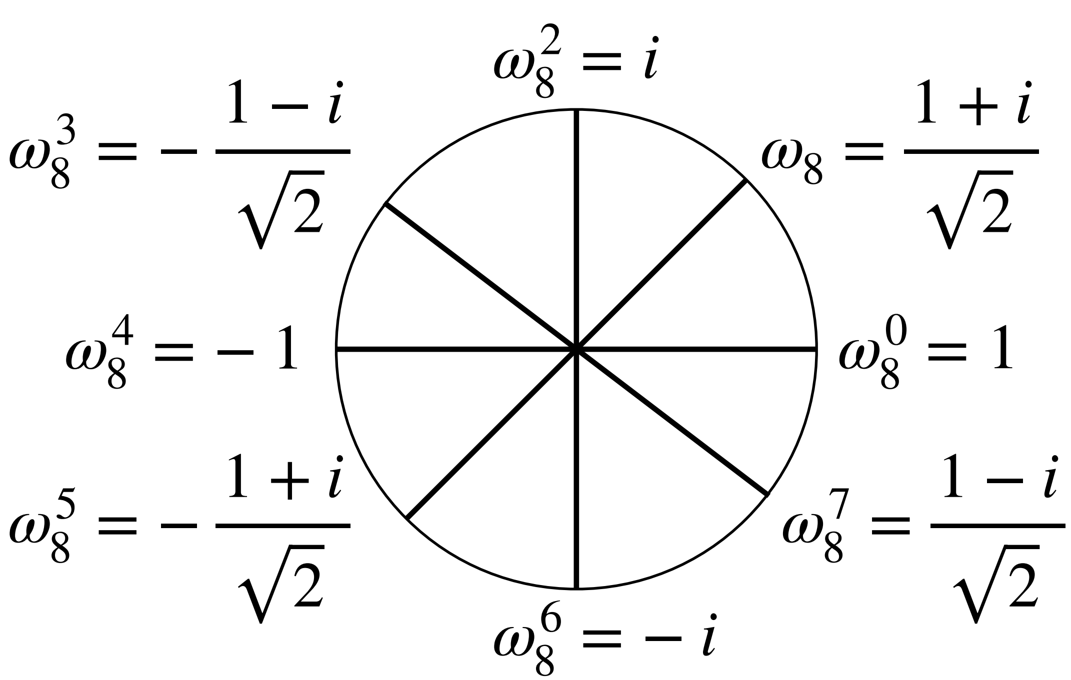
The quantity $e^{\frac{2\pi i}{n}} = \cos(2\pi/n) + i\sin(2\pi/n)$ is called the *principal nth root of unity.*

Let $\omega_n = \cos(2\pi/n) + i\sin(2\pi/n) = e^{\frac{2\pi i}{n}}$

The nth roots of unity can then be written as:

$$1, \omega_n, \omega_n^2, \omega_n^3, \ldots, \omega_n^{n-1}$$

since $e^{\frac{2\pi ki}{n}} = \left(e^{\frac{2\pi i}{n}}\right)^k = \omega_n^k$



Properties of the Roots of Unity

Cancellation: Let $n \geq 0, k > 0, d > 0$. It holds that $\omega_{dn}^{dk} = \omega_n^k$

Proof: $\omega_{dn}^{dk} = \left(e^{\frac{2\pi i}{dn}}\right)^{dk}$

$$\begin{array}{cccccccc}
1 & \omega_n^2 & \cdots & \omega_n^{n-2} & \omega_n^n & \omega_n^{n+2} & \cdots & \omega_n^{2(n-1)} \\
\| & \| & \cdots & \| & \| & \| & \cdots & \| \\
1 & \omega_{n/2} & \cdots & \omega_{n/2}^{n/2-1} & 1 & \omega_{n/2} & \cdots & \omega_{n/2}^{n/2-1}
\end{array}$$

Halving: Let $n > 0$ be even. Then if we square all the n nth roots of unity, we get all $n/2$ $(n/2)$th roots of unity, each one twice.

Proof: $\left(\omega_n^k\right)^2 = \omega_n^{2k} = \omega_{n/2}^k$, also

$\left(\omega_n^{k+n/2}\right)^2 = \omega_n^{2k+n} = \omega_n^{2k} \cdot \omega_n^n = \omega_{n/2}^k$

Properties of the Roots of Unity

Summation: Suppose $n \geq 1$ and k is not divisible by n. It holds that $\sum_{j=0}^{n-1} \left(\omega_n^k\right)^j = 0$

Proof: $$\sum_{j=0}^{n-1} \left(\omega_n^k\right)^j = \frac{\left(\omega_n^k\right)^n - 1}{\omega_n^k - 1} = \frac{\left(\omega_n^n\right)^k - 1}{\omega_n^k - 1} = \frac{1^k - 1}{\omega_n^k - 1} = 0$$

sum of geometric series

Running time

$\Omega(n)$ for each j

$\Omega(n^2)$ overall

Step 1: Choose $2n$ values $x_1, x_2, \ldots, x_{2n}$ and *evaluate* $A(x_j)$ and $B(x_j)$ for each $j = 1,2,\ldots,2n$.

Step 2: Compute $C(x_j) = A(x_j) \cdot B(x_j)$ for all j $O(n)$

(these are now just numbers).

Step 3: Recover C from $C(x_1), C(x_2), \ldots, C(x_{2n})$.

? no idea for now

We will choose the $2n$ values carefully!

Running time

$\Omega(n)$ for each j

$\Omega(n^2)$ overall

Step 1: Choose $2n$ values $x_1, x_2, \ldots, x_{2n}$ and *evaluate* $A(x_j)$ and $B(x_j)$ for each $j = 1,2,\ldots,2n$.

Step 2: Compute $C(x_j) = A(x_j) \cdot B(x_j)$ for all j $O(n)$

(these are now just numbers).

Step 3: Recover C from $C(x_1), C(x_2), \ldots, C(x_{2n})$.

? no idea for now

We will choose the $2n$th (complex) roots of unity.

Discrete Fourier Transform

The *Discrete Fourier Transform (DFT)* of a sequence of m complex numbers $p_0, p_1, \ldots, p_{m-1}$ is defined to be the sequence of complex numbers

$$P(1), P(\omega_m), P(\omega_m^2), \ldots, P(\omega_m^{m-1})$$

obtained by evaluating the polynomial

$$P(x) = p_0 + p_1 x + p_2 x^2 + \ldots, p_{m-1} x^{m-1}$$

on each of the mth roots of unity.

Divide and Conquer

Assume that $m = 2^\ell$ for some positive integer ℓ.

Let

$P_{\text{even}}(x) = p_0 + p_2 x + p_4 x^2 + \ldots + p_{m-2} x^{m/2-1}$

$P_{\text{odd}}(x) = p_1 + p_3 x + p_5 x^2 + \ldots + p_{m-1} x^{m/2-1}$

Observe that: $P(x) = P_{\text{even}}(x^2) + x \cdot P_{\text{odd}}(x^2)$

So to evaluate $P(x)$ at $1, \omega_m, \omega_m^2, \ldots, \omega_m^{m-1}$, we can

1. Evaluate the two polynomials of degree $m/2 - 1$ at $1^2, (\omega_m)^2, (\omega_m^2)^2, \ldots, (\omega_m^{m-1})^2$
2. Combine the results to obtain $P(x)$

Divide and Conquer

So to evaluate $P(x)$ at $1, \omega_m, \omega_m^2, \ldots, \omega_m^{m-1}$, we can

1. Evaluate the two polynomials of degree $m/2 - 1$ at $1^2, (\omega_m)^2, (\omega_m^2)^2, \ldots, (\omega_m^{m-1})^2$

We successfully halved the degree

It seems we are still evaluating on $m - 1$ points

2. Combine the results to obtain $P(x)$

Properties of the Roots of Unity

Cancellation: Let $n \geq 0, k > 0, d > 0$. It holds that $\omega^{dk} = \omega^{k}$

Proof: $\omega_{dn}^{dk} = \left(e^{\frac{2\pi i}{dn}}\right)^{dk}$

1	ω_n^2	$\ldots$	ω_n^{n-2}	ω_n^n	ω_n^{n+2}	$\ldots$	$\omega_n^{2(n-1)}$
$\Vert$	$\Vert$	$\ldots$	$\Vert$	$\Vert$	$\Vert$	$\ldots$	$\Vert$
1	$\omega_{n/2}$	$\ldots$	$\omega_{n/2}^{n/2-1}$	1	$\omega_{n/2}$	$\ldots$	$\omega_{n/2}^{n/2-1}$

Halving: Let $n > 0$ be even. Then if we square all the n nth roots of unity, we get all $n/2$ $(n/2)$th roots of unity, each one twice.

Proof: $\left(\omega_n^k\right)^2 = \omega_n^{2k} = \omega_{n/2}^k$, also

$$\left(\omega_n^{k+n/2}\right)^2 = \omega_n^{2k+n} = \omega_n^{2k} \cdot \omega_n^n = \omega_{n/2}^k$$

Divide and Conquer

So to evaluate $P(x)$ at $1, \omega_m, \omega_m^2, \ldots, \omega_m^{m-1}$, we can

1. Evaluate the two polynomials of degree $m/2 - 1$ at $1^2, (\omega_m)^2, (\omega_m^2)^2, \ldots, (\omega_m^{m-1})^2$
2. Combine the results to obtain $P(x)$

We successfully halved the degree

It seems we are still evaluating on m points

This is a list of the $m/2$ $(m/2)$th roots of unity, each appearing twice

So we only need to evaluate at $m/2$ points

Divide

$$\begin{array}{cccccccc} 1 & \omega_n^2 & \cdots & \omega_n^{n-2} & \omega_n^n & \omega_n^{n+2} & \cdots & \omega_n^{2(n-1)} \\ \| & \| & \cdots & \| & \| & \| & \cdots & \| \\ 1 & \omega_{n/2} & \cdots & \omega_{n/2}^{n/2-1} & 1 & \omega_{n/2} & \cdots & \omega_{n/2}^{n/2-1} \end{array}$$

So to evaluate $P(x)$ at $1, \omega_m, \omega_m^2, \ldots, \omega_m^{m-1}$, we can

1. Evaluate the two polynomials of degree $m/2 - 1$ at $1^2, (\omega_m)^2, (\omega_m^2)^2, \ldots, (\omega_m^{m-1})^2$
2. Combine the results to obtain $P(x)$

$$P(1) = P_{\textbf{even}}(1) + 1 \cdot P_{\textbf{odd}}(1)$$

$$P(\omega_m) = P_{\textbf{even}}(\omega_{m/2}) + \omega_m \cdot P_{\textbf{odd}}(\omega_{m/2})$$

$$P(\omega_m^2) = P_{\textbf{even}}(\omega_{m/2}^2) + \omega_m^2 \cdot P_{\textbf{odd}}(\omega_{m/2}^2)$$

$$\vdots$$

$$P(\omega_m^{m/2-1}) = P_{\textbf{even}}(\omega_{m/2}^{m/2-1}) + \omega_m^2 \cdot P_{\textbf{odd}}(\omega_{m/2}^{m/2-1})$$

$$P(\omega_m^{m/2}) = P(1)$$

$$P(\omega_m^{m/2+1}) = P(\omega_m)$$

$$\vdots$$

$$P(\omega_m^{m-1}) = P(\omega_m^{m/2-1})$$

Pseudocode (CLRS pp. 890)

FFT(a, n)

if $n == 1$
    return a // DFT of 1 element is the element itself
$\omega_n = e^{2\pi i/n}$
$\omega = 1$
$a^{\text{even}} = (a_0, a_2, \ldots, a_{n-2})$
$a^{\text{odd}} = (a_1, a_3, \ldots, a_{n-1})$
$y^{\text{even}} = \text{FFT}(a^{\text{even}}, n/2)$
$y^{\text{odd}} = \text{FFT}(a^{\text{odd}}, n/2)$
for $k = 0$ **to** $n/2 - 1$ // at this point, $\omega = \omega_n^k$
    $y_k = y_k^{\text{even}} + \omega\, y_k^{\text{odd}}$
    $y_{k+(n/2)} = y_k^{\text{even}} - \omega\, y_k^{\text{odd}}$
    $\omega = \omega\, \omega_n$
return y

Running time

Step 1: Choose $2n$ values $x_1, x_2, \ldots, x_{2n}$ and *evaluate* $A(x_j)$ and $B(x_j)$ for each $j = 1,2,\ldots,2n$.

Step 2: Compute $C(x_j) = A(x_j) \cdot B(x_j)$ for all j *(these are now just numbers).*

Step 3: Recover C from $C(x_1), C(x_2), \ldots, C(x_{2n})$.

Running time

Step 1: Choose the $2n$ $2n$th roots of unity $1, \omega_{2n}, \omega_{2n}^2, \ldots, \omega_{2n}^{2n-1}$ and *evaluate* $A(\omega_{2n}^j)$ and $B(\omega_{2n}^j)$ for each $j = 0,1,\ldots,2n-1$.

How much time do we need for each of the evaluations?

Let $T(n)$ be the time required to evaluate a polynomial of degree $n-1$ on all of the $2n$ $2n$th roots of unity.

We need to evaluate $P(x) = P_{\text{even}}(x^2) + x \cdot P_{\text{odd}}(x^2)$ at $1, \omega_{2n}, \omega_{2n}^2, \ldots, \omega_{2n}^{2n-1}$

Running time: $T(n) \le 2T(n/2) + cn$

Asymptotic running time: $O(n \log n)$

What if we divided like this?

Assume that $m = 2^\ell$ for some positive integer ℓ.

Let

$$P_{\text{small}}(x) = p_0 + p_1 x + p_2 x^2 + \ldots + p_{m/2-1} x^{m/2-1}$$
$$P_{\text{big}}(x) = p_{m/2} + p_{m/2+1} x + p_{m/2+2} x^2 + \ldots + p_{m-1} x^{m/2-1}$$

We would have: $P(x) = P_{\text{big}}(x) + x^{m/2} \cdot P_{\text{small}}(x)$

What is the issue with this?

Running time

$O(n \log n)$

Step 1: Choose the $2n$ $2n$th roots of unity $1, \omega_{2n}, \omega_{2n}^2, \ldots, \omega_{2n}^{2n-1}$ and *evaluate* $A(\omega_{2n}^j)$ and $B(\omega_{2n}^j)$ for each $j = 0, 1, \ldots, 2n-1$.

Step 2: Compute $C(x_j) = A(x_j) \cdot B(x_j)$ for all j *(these are now just numbers).*

$O(n)$

Step 3: Recover C from $C(x_1), C(x_2), \ldots, C(x_{2n})$.

What about this?

(Fast) Polynomial Interpolation

Recover C from $C(x_1), C(x_2), \ldots, C(x_{2n})$

Main idea: We will reduce *polynomial interpolation* to *polynomial evaluation*, which we saw how to do using D&C earlier.

Define the polynomial $D(x) = \sum_{s=0}^{2n-1} C(\omega_{2n}^{s}) \cdot x^s$, and evaluate it at the $2n$th roots of unity.

(Fast) Polynomial Interpolation

Define the polynomial $D(x) = \sum_{s=0}^{2n-1} C(\omega_{2n}^{s}) \cdot x^{s}$, and evaluate it at the $2n$th roots of unity.

$$
\begin{aligned}
D(\omega_{2n}^{k}) &= \sum_{s=0}^{2n-1} C(\omega_{2n}^{s}) \cdot \left(\omega_{2n}^{k}\right)^{s} \\
&= \sum_{s=0}^{2n-1} \left(\sum_{t=0}^{2n-1} c_t \left(\omega_{2n}^{s}\right)^{t} \right) \left(\omega_{2n}^{k}\right)^{s} \\
&= \sum_{t=0}^{2n-1} c_t \left(\sum_{s=0}^{2n-1} \left(\omega_{2n}^{s}\right)^{t} \left(\omega_{2n}^{k}\right)^{s} \right) = \sum_{t=0}^{2n-1} c_t \left(\sum_{s=0}^{2n-1} \omega_{2n}^{st+ks} \right)
\end{aligned}
$$

(Fast) Polynomial Interpolation

Define the polynomial $D(x) = \sum_{s=0}^{2n-1} C(\omega_{2n}^{s}) \cdot x^{s}$, and evaluate it at the $2n$th roots of unity.

$$D(\omega_{2n}^{k}) = \sum_{t=0}^{2n-1} c_t \left(\sum_{s=0}^{2n-1} \omega_{2n}^{st+ks} \right)$$

$$= \sum_{t=0}^{2n-1} c_t \left(\sum_{s=0}^{2n-1} \left(\omega_{2n}^{t+k}\right)^{s} \right)$$

Properties of the Roots of Unity

Summation: Suppose $n \geq 1$ and k is not divisible by n. It holds that $\sum_{j=0}^{n-1} \left(\omega_n^k\right)^j = 0$

Proof: $$\sum_{j=0}^{n-1} \left(\omega_n^k\right)^j = \frac{\left(\omega_n^k\right)^n - 1}{\omega_n^k - 1} = \frac{\left(\omega_n^n\right)^k - 1}{\omega_n^k - 1} = \frac{1^k - 1}{\omega_n^k - 1} = 0$$

sum of geometric series

(Fast) Polynomial Interpolation

Define the polynomial $D(x) = \sum_{s=0}^{2n-1} C(\omega_{2n}^{s}) \cdot x^{s}$, and evaluate it at the $2n$th roots of unity.

$$D(\omega_{2n}^{k}) = \sum_{t=0}^{2n-1} c_t \left(\sum_{s=0}^{2n-1} \omega_{2n}^{st+ks} \right)$$

$$= \sum_{t=0}^{2n-1} c_t \left(\sum_{s=0}^{2n-1} \left(\omega_{2n}^{t+k}\right)^{s} \right)$$

$$= c_{2n-k} \cdot 2n$$

For all t such that $t + k$ is not divisible by $2n$, we have:

$$\sum_{s=0}^{2n-1} \left(\omega_{2n}^{t+k}\right)^{s} = 0$$

When $t + k$ is divisible by $2n$, (i.e., when $t = 2n - k$) we have

$$\omega_{2n}^{t+k} = 1$$

(Fast) Polynomial Interpolation

Recover C from $C(x_1), C(x_2), \ldots, C(x_{2n})$

Define the polynomial $D(x) = \sum_{s=0}^{2n-1} C(\omega_{2n}^s) \cdot x^s$, and evaluate it at the $2n$th roots of unity.

We get: $c_s = \frac{1}{2n} \cdot D\left(\omega_{2n}^{2n-s}\right)$

Alternative viewpoint

The *Discrete Fourier Transform (DFT)* of a sequence of m complex numbers $p_0, p_1, \ldots, p_{m-1}$ is defined to be the sequence of complex numbers

$$P(1), P(\omega_m), P(\omega_m^2), \ldots, P(\omega_m^{m-1})$$

obtained by evaluating the polynomial

$$P(x) = p_0 + p_1 x + p_2 x^2 + \ldots, p_{m-1} x^{m-1}$$

on each of the mth roots of unity.

Alternative viewpoint

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_m & \omega_m^2 & \cdots & \omega_m^{m-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega_m^{m-1} & \omega_m^{2(m-1)} & \cdots & \omega_m^{(m-1)^2} \end{bmatrix}}_{M} \begin{bmatrix} p_0 \\ p_1 \\ \\ p_{m-1} \end{bmatrix} = \begin{bmatrix} \hat{p}_0 \\ \hat{p}_1 \\ \\ \hat{p}_{m-1} \end{bmatrix}$$

We can compute $\overrightarrow{p} = M^{-1}\overrightarrow{\hat{p}}$

Is M invertible?

How can we compute M^{-1}?

M is invertible

$$\begin{bmatrix} 1 & z_0 & z_0^2 & \cdots & z_0^m \\ 1 & z_1 & z_1^2 & \cdots & z_1^m \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & z_\ell & z_\ell^2 & \cdots & z_\ell^m \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \\ p_{m-1} \end{bmatrix} = \begin{bmatrix} \hat{p}_0 \\ \hat{p}_1 \\ \\ \hat{p}_{m-1} \end{bmatrix}$$

Vandermonde matrix

$$\mathbf{det}(M) = \prod_{0 \le i < j \le \ell} (x_j - x_i)$$

When $m = \ell$ (i.e., M is square) and $z_i \neq z_j$ for all $i \neq j$ (i.e., all z_i's are distinct and thus $\mathrm{det}(M) \neq 0$, then M is invertible.

How to compute M

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_m & \omega_m^2 & \cdots & \omega_m^{m-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega_m^{m-1} & \omega_m^{2(m-1)} & \cdots & \omega_m^{(m-1)^2} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \\ p_{m-1} \end{bmatrix} = \begin{bmatrix} \hat{p}_0 \\ \hat{p}_1 \\ \\ \hat{p}_{m-1} \end{bmatrix}$$

$$\underbrace{}_{M(\omega_m)}$$

Lemma: $M(\omega_m)^{-1} = \frac{1}{m} M(\omega_m^{-1})$

How to compute M

Lemma: $M(\omega_m)^{-1} = \frac{1}{m} M(\omega_m^{-1})$

Proof: $M(\omega_m)(j, j') = \omega_m^{jj'}$, and $\frac{1}{m} M(\omega_m^{-1})(j, j') = \frac{1}{m} \omega_m^{-jj'}$

Consider the matrix $\frac{1}{m} M(\omega_m^{-1}) \cdot M(\omega_m)$

Then we have:

$$\frac{1}{m} M(\omega_m^{-1}) \cdot M(\omega_m)(j, j') = \frac{1}{m} \sum_{k=0}^{m-1} \omega_m^{-kj} \cdot \omega_m^{kj'} = \frac{1}{m} \sum_{k=0}^{m-1} \omega_m^{k(j'-j)}$$

How to compute M

Then we have:

$$\frac{1}{m}M(\omega_m^{-1}) \cdot M(\omega_m)(j,j') = \frac{1}{m}\sum_{k=0}^{m-1}\omega_m^{-kj}\cdot\omega_m^{kj'} = \frac{1}{m}\sum_{k=0}^{m-1}\omega_m^{k(j'-j)}$$

If $j = j'$, then $\frac{1}{m}M(\omega_m^{-1}) \cdot M(\omega_m)(j,j') = 1$

If $j \neq j'$, then $\frac{1}{m}\sum_{k=0}^{n-1}\omega_m^{k(j'-j)} = 0$ by *summation*.

Why? Because $-(m-1) \leq j' - j \leq m - 1$

How to compute M

Lemma: $M(\omega_m)^{-1} = \frac{1}{m} M(\omega_n^{-1})$

Hence $\frac{1}{m} M(\omega_m^{-1}) \cdot M_m(\omega_m) = I_m$ (the identify matrix).

Running time

$O(n \log n)$

Step 1: Choose the $2n$ $2n$th roots of unity $1, \omega_{2n}, \omega_{2n}^2, \ldots, \omega_{2n}^{2n-1}$ and *evaluate* $A(\omega_{2n}^j)$ and $B(\omega_{2n}^j)$ for each $j = 0,1,\ldots,2n-1$.

Step 2: Compute $C(x_j) = A(x_j) \cdot B(x_j)$ for all j *(these are now just numbers).*

$O(n)$

Step 3: Recover C from $C(x_1), C(x_2), \ldots, C(x_{2n})$.

What about this?

Running time

$O(n \log n)$

Step 1: Choose the $2n$ $2n$th roots of unity $1, \omega_{2n}, \omega_{2n}^2, \ldots, \omega_{2n}^{2n-1}$ and *evaluate* $A(\omega_{2n}^j)$ and $B(\omega_{2n}^j)$ for each $j = 0,1,\ldots,2n-1$.

Step 2: Compute $C(x_j) = A(x_j) \cdot B(x_j)$ for all j *(these are now just numbers).* $O(n)$

Step 3: Recover C from $C(x_1), C(x_2), \ldots, C(x_{2n})$.

$O(n \log n)$

Convolution Theorem

For any two vectors a and b of length n where n is a power of 2, the convolution $a * b$ of a and b can be computed as:

$$a * b = \mathsf{DFT}_{2n}^{-1}\left(\mathsf{DFT}_{2n}(a) + \mathsf{DFT}_{2n}(b)\right)$$