Multiplying Polynomials

Suppose we have two (uni-variate) polynomials

$A(x) = 3x^2 - 5x + 7$

$B(x) = -2x^2 + x - 4$

↑ fancy name for "having one variable"

What is $A(x) \cdot B(x)$?

$$
\begin{array}{rrrrr}
 & & 3x^2 & -5x & +7 \\
 & & -2x^2 & +x & -4 \\
\hline
 & & -12x^2 & +20x & -28 \\
 & 3x^3 & -5x^2 & +7x & \\
-6x^4 & +10x^3 & -14x^2 & & \\
\hline
-6x^4 & +13x^3 & -31x^2 & +27x & -28.
\end{array}
$$

If $A(x)$ and $B(x)$ have degree (at most) n,

$A(x) = a_0 + a_1 x + \dots + a_n x^n = \sum_{i=0}^{n} a_i \cdot x^i$

$B(x) = b_0 + b_1 x + \dots + b_n x^n = \sum_{i=0}^{n} b_i \cdot x^i$

each is "represented" by $n+1$ coefficients and multiplication takes $O(n^2)$ time.

Can we do better?

similar to integer multiplication...

Point-value representation. Suppose that $A(x)$ is a polynomial of degree at most n, and we know

$$\begin{cases} A(x_0) = y_0 \\ \vdots \\ A(x_n) = y_n \end{cases} \quad \text{for } n+1 \text{ distinct } x_0, \dots, x_n. \quad (*)$$

Then, do we "know" $A(x)$? More precisely, is there a unique polynomial $A(x) = a_0 + a_1 x + \cdots + a_n x^n$ satisfying them?

Existence. $A(x) = \sum_{i=0}^{n} y_i \cdot \dfrac{\prod_{j \neq i} (x - x_j)}{\prod_{j \neq i} (x_i - x_j)}$ satisfies $(*)$.

$= \begin{cases} 0 & \text{if } x = x_j \text{ for some } j \in \{0, \dots, n\} \setminus i. \\ 1 & \text{if } x = x_i. \end{cases}$

Uniqueness. If $A(x), B(x)$ both satisfy $(*)$, then

$A(x_i) - B(x_i) = 0 \quad \forall i \in \{0, \dots, n\}.$

It is a degree-n polynomial with $n+1$ zeros, so should be the zero polynomial.

So, $(x_0, A(x_0)), \dots, (x_n, A(x_n))$ is another way to represent a degree-n polynomial $A(x)$.

If $A(x), B(x)$ are degree-n polynomials represented as $(x_0, A(x_0)), \dots, (x_{2n}, A(x_{2n}))$ and $(x_0, B(x_0)), \dots, (x_{2n}, B(x_{2n}))$, and $C(x) := A(x) \cdot B(x)$, then $(x_0, A(x_0)B(x_0)), \dots, (x_{2n}, A(x_{2n})B(x_{2n}))$ represents $C(x)$, so we multiplied A and B in linear time!

Can we use this even when A, B are given by coefficients?

First step: given n-degree polynomial $A(x) = \sum_{i=0}^{n} a_i \cdot x^i$, compute
$(x_0, A(x_0)), \ldots, (x_{2n}, A(x_{2n}))$ at different $x_0, \ldots x_{2n}$.

(actually, treat $A(x)$ as degree-$2n$ polynomial so that degree = (# evaluations we want).)

Again, naively, $\Theta(n)$ evaluations, each of which takes $\Theta(n)$ times $\longrightarrow \Theta(n^2)$ total.

But, we have freedom to choose $x_0, \ldots, x_{2n}$.

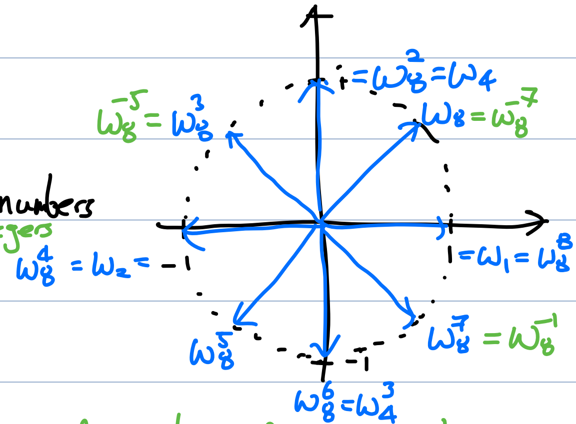
So each x_i doesn't need to be computed "from scratch"
What choices of $x_0, \ldots, x_{2n}$ are good?

Complex roots of unity

Define $\omega_n = e^{2\pi i/n} = \cos(2\pi/n) + i\cdot\sin(2\pi/n) \in \mathbb{C}$

$\sqrt{-1}$.

set of complex numbers



Facts,

① $\omega_n^1,\dots,\omega_n^n$ are n different numbers

② $(\omega_n^k)^n = 1 \quad \forall k\in\mathbb{Z}$. ← set of integers $\{\dots,-1,0,+1,\dots\}$

③ $\omega_{dn}^{dk} = \omega_n^k \quad \forall n,k,d\in\mathbb{N}$.

set of natural numbers $\{1,2,3,\dots\}$

④ If k is not a multiple of n,

$$\sum_{j=0}^{n-1}(\omega_n^k)^j = 0.$$

Fast Fourier Transform.

__Definition,__ Given n-dimensional vector $(a_0, \dots, a_{n-1}) \in \mathbb{C}^n$
(which represents polynomial $A(x) = \sum_{j=0}^{n-1} a_j x^j$),
its Discrete Fourier Transform (DFT) is $(y_0 \dots y_{n-1}) \in \mathbb{C}^n$
defined by $y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j (\omega_n^k)^j = \sum_{j=0}^{n-1} a_j \cdot e^{-2\pi i (kj/n)}$

The "standard" Fourier Transform, given $f : \mathbb{R} \to \mathbb{C}$,
outputs $\hat{f} : \mathbb{R} \to \mathbb{C}$ defined by $\hat{f}(\alpha) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i \alpha x} dx$.

(D)FT is important in signal processing, math/physics, ...

Fast Fourier Transform (FFT) : $O(n \log n)$-time algorithm to compute it.

Will use Divide and Conquer
– How to "split the problem into two?"

(assume n even)

$A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \dots + a_{n-1} x^{n-1}$

$A_0(x) = a_0 + a_2 x^1 + a_4 x^2 \dots + a_{n-2} x^{n/2 - 1}$

$A_1(x) = a_1 + a_3 x + \dots + a_{n-1} x^{n/2-1}$,

(Formally, $A_0(x) = \sum_{j=0}^{n/2-1} a_{2j} \cdot x^j$

$A_1(x) = \sum_{j=0}^{n/2-1} a_{2j+1} \cdot x^j$

Simple but crucial magic: $A(x) = A_0(x^2) + x \cdot A_1(x^2)$!!

	A	A_0, A_1
degree	$n-1$	$n/2 - 1$
want to evaluate at	$\omega_n^0, \dots, \omega_n^{n-1}$	$\omega_{n/2}^0, \omega_{n/2}^1, \dots, \omega_{n/2}^{n-1}$
		$= \omega_n^0, \omega_n^2, \dots, \omega_n^{n-2}$
		(by fact ③)

So, if we evaluate A_0, A_1 at $\omega_{n/2}, \dots, \omega_{n/2}^{n-1}$, for any $j \in \{0..n-1\}$,

$$A(\omega_n^j) = \underbrace{A_0(\omega_n^{2j})}_{\substack{= A_0(\omega_{n/2}^{j}) \\ = A_0(\omega_{n/2}^{j \bmod n/2})}} + \omega_n^j \cdot \underbrace{A_1(\omega_n^{2j})}_{\substack{= A_1(\omega_{n/2}^{j}) \\ = A_1(\omega_{n/2}^{j \bmod n/2})}}$$

can be computed in $O(1)$ time! So total "merge" time $= O(n)$.

Recursive-FFT(a, n) (n is power of 2, $a = (a_0 \dots a_{n-1}) \in \mathbb{C}^n$)

$b = (a_0, a_2, \dots, a_{n-2})$ $\quad c = (a_1, a_3, \dots, a_{n-1})$

$s = $ Recursive-FFT(b) $\quad t = $ Recursive-FFT(c)

$\omega = 1$

For $k = 0$ to $n-1$

  $y_k = s_{(k \bmod n/2)} + \omega \cdot t_{(k \bmod n/2)}$

  $\omega = \omega \cdot \omega_n$.

Return y

Running time. Breaks a problem of size n into

* $k=2$ smaller problems
* each with size $n/b = n/2$
* with cost $O(n^d) = O(n)$ to combine.

Master Theorem with $k=b=2$, $d=1$ gives $O(n \log n)$.

Finishing Polynomial Mult.

Given degree-n polynomials $A(x), B(x)$. Want to compute $C(x) = A(x) \cdot B(x)$.
Let $m \in \mathbb{N}$ s.t. (1) $2n+1 \le m \le 4n$, (2) m is power of 2.

Using FFT, computed $A(\omega_m^j), B(\omega_m^j)$ for $j \in \{0, \dots, m-1\}$
Then, easy to compute $C(\omega_m^j)$ for $j \in \{0, \dots, m-1\}$.
Since degree of $C \le 2n < m$, C already "determined".
But how to compute coeffients of C?

Let $c \in \mathbb{C}^m$ s.t. $C(x) = \sum_{j=0}^{m-1} c_j x^j$.
Let $y \in \mathbb{C}^n$ s.t. $C(\omega_m^k) = y_k = \sum_{j=0}^{m-1} c_j \omega_m^{kj}$ $\forall k \in \{0, \dots, m-1\}$
$(y = \mathrm{FFT}(c))$.

<u>**Lemma.**</u> $\forall j \in \{0, \dots, m-1\}$, $c_j = \frac{1}{m} \cdot \sum_{k=0}^{m-1} y_k \cdot \omega_m^{-kj}$.

<u>Pf</u>. Fix j. For each $k \in \{0, \dots, m-1\}$, consider
$y_k = \sum_\ell c_\ell \omega_m^{k\ell} \Rightarrow \omega_m^{-kj} \cdot y_k = \sum_\ell c_\ell \omega_m^{k(\ell - j)}$ and sum m equations.

$$\sum_k \omega_m^{-kj} y_k = \sum_\ell c_\ell \left(\sum_k \omega_m^{k(\ell-j)} \right) = m \cdot c_j \quad \text{(by Fact 4)}$$

□

So, just replacing ω_m by ω_m^{-1}, c can be obtained from y via another FFT!