Lec 2 - Multiplying Polynomials

Recall → multiplying integers for D&Q using Karatsuba

Int mult to Poly mult

int $a = a_{n-1} a_{n-2} \cdots a_0$ (binary) → $A(x) = a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$

note: $A(0) = a_0$ (odd or even)

$A(1) = \sum_i a_i$ (sum of coeff) hamming weight (no of 1s)

$A(2) = a$

efficient way to eval → $(\cdots (a_{n-1}x) + a_{n-2})x + a_{n-3})x \cdots + a_1)x + a_0$

Cs horner's rule!

Now consider a binary poly B, deg $n-1$. Find $C = BA$

note $C = C(2) = A(2)B(2)$

Trivial way to get C coeff

→ for i from 0 to $2(n-1)$

$c_i = 0$;

for j from 0 to i

$c_i \mathrel{+}= a_j b_{i-j}$ (convolution)

} $O(n^2)$

Karatsuba Approach

Chop $A(x)$ (deg $n-1$) into 2 of $(n/2 - 1)$.

$A(x) = x^{n/2} A_{hi}(x) + A_{lo}(x)$

$AB = x^n A_{hi} B_{hi} + x^{n/2}(A_{hi} B_{lo} + B_{hi} A_{lo}) + A_{lo} B_{lo}$

Recursive → $T(n) = 4T(n/2) + n$ → $O(n^2)$

Karatsuba

Co $T(n) = 3T(n/2) + n$ → $O(n^{\log_2 3})$

$= O(n^{1.58\ldots})$

Representations

eg Binary → addition is easy. Mult is hard
Prime Factorization → mult is easy!

e.g. Complex numbers
— Represent: cartesian → a+bi → easy to add. Okay to mult
Polar → $re^{i\theta}$ → hard " " easy " mult

E.g. Degree n-1 poly
↳ coeff list → (2,3,4) = $2x^2+3x+4$ → easy to add
↳ mult is $O(n^2)$ trivial, $O(n^{\log_2 3})$ known

↳ point-value rep → p is a degree-(n-1) poly, p is uniquely det by n eval $(p(x_0), \dots, p(x_{n-1}))$
↳ addition easy at the points
↳ mult easy at the points
↳ hard to evaluate outside the points

Game plan

coeff list ① → A → $((w_0, A(w_0)), \dots (w^{n-1}, A(w^{n-1}))$
B → ⋯⋯

② ↳ multiply to get C coeff list ③ → convert to coeff list for C

But! we have n vals for deg 2n poly. So make $a = (a_0, \dots, a_{n-1}, 0, \dots 0)$ & eval at 2n-1 places!

① $A(x) = a_0 + a_1x + a_2x^2 + a_3x^3$ —split→ $A_{even} = a_0 + a_2x$, $A_{odd} = a_1 + a_3x$

a_n { $a_n x^{n/2}$ 2|n ; $a_n x^{\lfloor n/2 \rfloor}$ 2∤n

Lets say we know $A_{odd}(25)$ & $A_{even}(25)$. Can we get A(5)?
we get $A(5) = A_{odd}(5^2)\cdot 5 + A_{even}(5^2)$!
Also $A(-5) = A_{odd}((-5)^2)\cdot(-5) + A_{even}((-5)^2)$ for free!

idea: A on $\{5, -5, -1, 1, -4, 4, \dots\}$ n
↓ D2Q
Aodd, Aeven $\{25, \dots, 16, \dots\}$ n/2

but not n/4? No pairs. But! ☹

we want a list that halfs each time. Roots of unity!

let ω_n be the n^{th} root of unity in $\mathbb{C}$ one counterclockwise turn from 1!

list is $\{\omega_n, \omega_n^2, \ldots, \omega_n^n = 1\}$

note $\omega_n^{k+n/2} = -\omega_n^k$

$\Rightarrow (\omega_n^{k+n/2})^2 = (\omega_n^k)^2$

halfing the list!

The Fast Fourier Transform

- A is a list of n coeff. n a power of 2
- FFT$(a_0, a_1, \ldots a_{n-1}, \omega_n)$ → eval A on $\{\omega_n^i\}_{0 \le i < n}$
  - if $n == 1$ return a_0 (base) → eval of A(1) is a_0
  - $(d_0, \ldots, d_{n/2-1}) =$ FFT$(a_0, a_2, \ldots a_{n-2}, (\omega_n)^2)$
  - $(e_0, \ldots, e_{n/2-1}) =$ FFT$(a_1, a_3, \ldots a_{n-1}, (\omega_n)^2)$
  - For k from 0 to n/2 − 1
    - $y_k = d_k + (\omega_n)^k e_k$
    - $y_{k+n/2} = d_k - (\omega_k)^k e_k$
  - ret $(y_0, \ldots, y_{n-1})$

note: for $k \le n/2$

$A(\omega_n^k) = Ae((\omega_n^2)^k) + \omega_n^k Ao((\omega_n^2)^k)$

$A(\omega_n^{k+n/2}) = Ae(\ldots) - \omega_n^k Ao(\ldots)$

2 for 1

FFT Is Actually

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & (\omega_n)^2 & \cdots & (\omega_n)^{n-1} \\ 1 & (\omega_n^2) & (\omega_n^2)^2 & \cdots & (\omega_n^2)^{n-1} \\ \vdots & & & \ddots & \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix}$$

its actually matrix mult!

$HA = Y$

to get coeff - Get H^{-1} to inv!

$$H^{-1} = \frac{1}{n} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^{-1} & (\omega_n^{-1})^2 & \cdots & (\omega_n^{-1})^{n-1} \\ 1 & (\omega_n^{-2}) & (\omega_n^{-2})^2 & & \\ \vdots & & & \ddots & \\ 1 & \omega_n^{-(n-1)} & \cdots & & \end{bmatrix}$$

So IFFT$(y_0, y_1, \ldots, y_{n-1}, \omega_n) = \frac{1}{n}FFT(y_0, y_1, \ldots, y_{n-1}, \omega_n^{-1})$