COL 351 Lecture 6 2023/01/16

Topic : Faster Polynomial Multiplication via Fast Fourier Transform

Announcement Tutorial Slots

Wed 14:00 – 15:00 } LH 316 (?)
Thu 18:30 – 19:30 }

Polynomial Multiplication

Input: $A(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{d_1 - 1}$ $A = [a_0 \ldots a_{d_1-1}]$

$B(x) = b_0 + b_1 x + \ldots + b_{n-1} x^{d_2 - 1}$ $B = [b_0 \ldots b_{d_2-1}]$

Output: $A(x) \cdot B(x) = C(x)$ (say) $C = [c_0 \ldots c_{d_1+d_2-2}]$

$c_i = a_0 b_i + a_1 b_{i-1} + \ldots + a_i b_0$

Last lecture: Karatsuba's Algorithm — $O(n^{\log_2 3})$ time

Claim: For every n, and complex numbers $x_0 \dots x_{n-1}$, $y_0 \dots y_{n-1}$ such that $x_0 \dots x_{n-1}$ are distinct, among the polynomials with deg $\leq n-1$, $\exists$ a unique polynomial p such that $p(x_i) = y_i \;\; \forall\, i = 0 \dots n-1$

Proof sketch: Suppose $p(x) = p_0 + p_1 x + \dots + p_{n-1} x^{n-1}$

$$\forall i \;\; p(x_i) = y_i \iff \begin{bmatrix} x_0^0 & x_0^1 & \dots & x_0^{n-1} \\ x_1^0 & \dots & \dots & x_1^{n-1} \\ \vdots & \ddots & \vdots & \vdots \\ x_{n-1}^0 & \dots & \dots & x_{n-1}^{n-1} \end{bmatrix} \begin{bmatrix} p_0 \\ \vdots \\ p_{n-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ \vdots \\ y_{n-1} \end{bmatrix}$$

"Vandermonde Matrix"

Is invertible iff $x_0, \dots, x_{n-1}$ distinct.

Polynomial Multiplication Algorithm Plan

$$\underbrace{A(x)}_{\text{deg } d_1-1} \qquad \underbrace{B(x)}_{d_2-1} \xrightarrow{\text{want}} \underbrace{C(x)}_{\text{deg } d_1+d_2-2} = A(x)\,B(x)$$

$\downarrow$ how? today $\qquad\qquad\qquad \uparrow$ how? next class.

$$\begin{matrix} A(x_0) & B(x_0) & & C(x_0) = A(x_0)\,B(x_0) \\ \vdots & \vdots & \xrightarrow[O(n)]{\text{easy!}} & \vdots \\ A(x_{n-1}) & B(x_{n-1}) & & C(x_{n-1}) = A(x_{n-1})\,B(x_{n-1}) \end{matrix}$$

$n \geq d_1 + d_2 - 1$ $\qquad$ n: power of 2 $\qquad$ $R_n = \{x_0 \dots x_{n-1}\}$

Input: $A(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$ $\quad$ n: power of 2

$A = [a_0, a_1, \ldots, a_{n-1}]$ $\quad$ $R_n = \{x_0 \ldots x_{n-1}\}$

Output: $A(x_0), A(x_1) \ldots A(x_{n-1})$

$A_0 = [a_0\ a_2 \ldots, a_{n-2}]$ $\quad$ $A_1 = [a_1, a_3 \ldots, a_{n-1}]$

$A_0(x) = a_0 + a_2 x + \cdots + a_{n-2} x^{\frac{n}{2}-1}$ $\quad$ $A_1(x) = a_1 + a_3 x + \cdots + a_{n-1} x^{\frac{n}{2}-1}$

$A(x) = A_0(x^2) + x A_1(x^2)$ $\quad$ Want: $A(x)\ \forall x \in R_n$

Desirable property: $R_{n/2}$ contains the square of every number in R_n. ie. $R_{n/2} = \{x^2 \mid x \in R_n\}$

$R_1 = \{1\}$ $\quad$ $R_2 = \{1, -1\}$ $\quad$ $R_4 = \{1, i, -1, -i\}$

$R_n = \{\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}\}$ $\quad$ $\omega_n = e^{\frac{2\pi}{n} i}$

Discrete Fourier Transform Problem (DFT)

Input: $A(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$ $\quad$ n: power of 2

$A = [a_0, a_1, \ldots, a_{n-1}]$

Output: $A(\omega_n^0), A(\omega_n^1) \ldots A(\omega_n^{n-1})$ where $\omega_n = e^{\frac{2\pi}{n} i}$

$(R_n = \{\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}\})$

$A_0 = [a_0\ a_2 \ldots, a_{n-2}]$ $\quad$ $A_1 = [a_1, a_3 \ldots, a_{n-1}]$

$A(x) = A_0(x^2) + x A_1(x^2)$

$\therefore A(\omega_n^i) = A_0(\omega_n^{2i}) + \omega_n^i A_1(\omega_n^{2i}) = A_0(\omega_{n/2}^i) + \omega_n^i A_1(\omega_{n/2}^i)$

Divide-and-Conquer Algorithm for DFT:

a.k.a. Fast Fourier Transform (FFT) Algorithm [Cooley–Tukey, 1965]

1. Construct A_0, A_1 —— $O(n)$

2. Recursively find DFT of A_0, DFT of A_1. —— $T(\frac{n}{2}) + T(\frac{n}{2})$

   This gives $A_0(\omega_{n/2}^i)$, $A_1(\omega_{n/2}^i)$ $\forall\, i = 1 \ldots n/2 - 1$

3a. For $i = 0 \ldots \frac{n}{2} - 1$:

$$A(\omega_n^i) = \underline{A_0(\omega_{n/2}^i)} + \omega_n^i\, \underline{A_1(\omega_{n/2}^i)} \quad \text{—— } O(n)$$

3b. For $i = n/2 \ldots n-1$

$$A(\omega_n^i) = A_0(\omega_{n/2}^i) + \omega_n^i\, A_1(\omega_{n/2}^i)$$

because $\omega_{n/2}^{n/2} = 1$ → $= \underline{A_0(\omega_{n/2}^{i-n/2})} + \omega_n^i\, \underline{A_1(\omega_{n/2}^{i-n/2})}$ —— $O(n)$

available from rec. calls.

4. Return $A(\omega_n^0), \ldots, A(\omega_n^{n-1})$

$T(n) = 2T(n/2) + cn \qquad \therefore\ T(n) = O(n \log n)$