Lecture 3, Jan 12

Plan for today:

1) Continue with matrix multiplication
2) Polynomial Multiplication.

Announcements: 1) Office hours for TAs 2-3:30 on Thursdays

2) Problem set 2 is up - to be discussed on Fri, Jan 21

Matrix Multiplication

Input: X, Y $n \times n$ matrices over integers

Output: $\underline{X \cdot Y}$

Naive: n^3 operations.

Divide and conquer:

$$X = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$$

$$Y = \left[\begin{array}{c|c} E & F \\ \hline G & H \end{array}\right]$$

$$X \cdot Y = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

$$= \left[\begin{array}{c|c} AE + BG & AF + BH \\ \hline CE + DG & CF + DH \end{array}\right]$$

Recurrence: $T(n) \leq \underline{8} \cdot T(\underline{n/2}) + O(n^2)$

Master theorem: $T(n) \leq O(n^3)$

Strassen: Reduce $n \times n$ Matrix Mult to $\underline{7}$ instances of $n/2 \times n/2$ Matrix Mult $+ O(n^2)$ pre and processing

Recurrence: $T(n) \leq 7 \cdot T(n/2) + O(n^2)$

Master theorem: $T(n) \leq n^{\log_2 7} = n^{\underline{2.81}}$

Faster algo. known: $\sim n^{\underline{2.3\ldots}}$

Strassen's subproblems

$X = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \quad Y = \begin{pmatrix} E & F \\ G & H \end{pmatrix}$

$P_1 = A \cdot (F - H)$

$P_2 = (A + B) H$

$P_3 = (C + D) \cdot E$

$P_4 = D (G - E)$

$P_5 = (A + D)(E + H)$

$P_6 = (B - D)(G + H)$

$P_7 = (A - C)(E + F)$

subproblems

$$X \cdot Y = \left(\begin{array}{c|c} AE + BG & AF + BH \\ \hline CE + DG & CF + DH \end{array} \right)$$

$$= \left(\begin{array}{c|c} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ \hline P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{array} \right) \quad (*)$$

combining subproblems

Algo:

0) Handle 1x1 matrices directly.

1) Break X, Y into $n/2 \times n/2$ blocks $A, B \ldots$

2) compute $n/2 \times n/2$ products $P_1, P_2, \ldots, P_7$

2) Compute $\frac{n}{2} \times \frac{n}{2}$ products $P_1, P_2, \ldots, P_7$ recursively

3) Combine the solution using Eqn (*).

Complexity. $T(n) \leq 7T(n/2) + O(n^2)$

$T(1) \leq 5$.

Correctness:

Assume: $n = 2^k$, Matrices are square.

- will assume this for Runtime analysis and correctness

What about $n \times n$ matrices when $n \neq 2^k$?

Idea: $\tilde{n} \times \tilde{n}$ Matrices

$\tilde{n} \neq 2^k$

- Want to view $\tilde{n} \times \tilde{n}$ Matrix Mult as $n \times n$ $M \cdot M$ for $n = 2^{k+1}$.

$n \le 2\tilde{n}$ – small constant factor increase in input size

- $$\underbrace{\begin{bmatrix} X & 0 \\ 0 & 0 \end{bmatrix}}_{X_1}^{\;n \times n} \cdot \underbrace{\begin{bmatrix} Y & 0 \\ 0 & 0 \end{bmatrix}}_{Y_1}^{\;n \times n}$$

$\tilde{n} \times \tilde{n}$

claim: To multiply x, y, suffices to multiply X_1, Y_1.

Pf:

$$X_1 \cdot Y_1 = \begin{bmatrix} X \cdot Y & 0 \\ 0 & 0 \end{bmatrix}_{n \times n}$$

Number of operations $\le O(\underline{n}^{\log_2 7}) \le O((2\tilde{n})^{\log_2 7})$
$\le \underline{O}(\tilde{n}^{\log_2 7})$

If $n \le \tilde{n}^{10}$

$\le O(\tilde{n}^{10 \cdot \log_2 7})$

Naive algo does better.

Back to correctness

Claim: $\forall\, k \in \mathbb{N} \cup \{0\}$ and matrices X, Y of size $2^k \times 2^k$ over integers ($\mathbb{Z}$), Strassen's algo correctly outputs the product $X \cdot Y$.

Pf: Induction on k.

Base case: $k = 0$ — step 0 of the algo.

Induction step:

Assume that the algo is correct for $2^{k-1} \times 2^{k-1}$ matrices. Will prove correctness for $2^k \times 2^k$ matrices.

From ind. hypo: $P_1, P_2, \ldots, P_7$

Suffices to show: combine the subproblems correctly.

(Eqn (*))

<u>Want to</u> <u>show</u>:

$$\begin{pmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{pmatrix} = \left(\begin{array}{c|c} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ \hline - - & - \end{array}\right)$$

① $AF + BH = P_1 + P_2$

<u>Pf</u>: $\underbrace{A(F-H)}_{P_1} + \underbrace{(A+B)\cdot H}_{P_2} = AF + BH$ ✓

(2) $\underline{A}E + \underline{BG} = P_5 + P_4 - P_2 + P_6$

Pf: RHS

$$(A+D)(E+H) + D(G-E) - (A+B)H + (B-D)(G+H)$$

$$= \underline{AE} + \cancel{AH} + \cancel{DE} + \cancel{DH}$$

$$+ \cancel{DG} - \cancel{DE}$$

$$- \cancel{AH} - \cancel{BH}$$

$$+ \underline{BG} + \cancel{BH} - \cancel{DG} - \cancel{DH}$$

HW: verify the remaining two identities. $\square$

Fast Polynomial Multiplication.

Input: $f = f_0 + f_1 x + f_2 x^2 + \cdots + f_{n-1} x^{n-1}$

$g = g_0 + g_1 x + g_2 x^2 + \cdots + g_{n-1} x^{n-1}$

Two poly. $f, g \in \mathbb{C}[x]$ — complex coefficients

given as a coeff vector (univariate)

$\deg \le n-1$

Output: coefficient vector of $f \cdot g$

Model: will only count the number of field operations.

Adding polynomials $f = f_0 + f_1 x + \cdots + f_{n-1} x^{n-1}$

$g = g_0 + g_1 x + \cdots + g_{n-1} x^{n-1}$

$$(f+g) = (f_0+g_0) + (f_1+g_1)x + (f_2+g_2)x^2 + \cdots + (f_{n-1}+g_{n-1})x^{n-1}.$$

$O(n)$ - operations.

Multiplication:

$$f \cdot g = u_0 + u_1 x + u_2 x^2 + \cdots + u_{2(n-1)} x^{2(n-1)}$$

$u_0 = f_0 g_0$

$u_1 = f_0 g_1 + f_1 g_0$

$u_2 = f_0 g_2 + f_2 g_0 + f_1 g_1$

$\vdots$

$$u_i = \sum_{j=0}^{i} f_j \cdot g_{i-j}$$

Naive algorithm:

Naive algorithm.

\# operations needed for $a_i \le 2i$

Total operations $\le \sum_{i=0}^{2n-2} 2i$

$= O(n^2)$.

Our plan: Algorithm that takes $O(n \log n)$ operations.

{Fast Fourier Transform (FFT)

– Cooley-Tukey 60s.

Representation of polynomials.

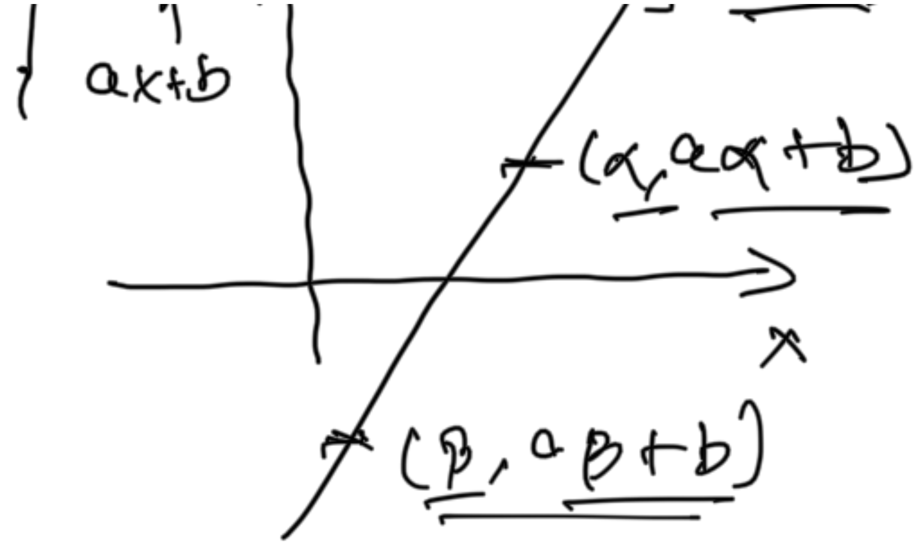
Natural: as a list of coefficients.

$ax+b$.

Another way:

$y=ax+b$.

By evaluations



$\alpha \neq \beta$

Lemma:

Interpolation

Let $\alpha_1, \alpha_2, \ldots, \alpha_n$ be distinct complex numbers.
Then, for any polynomial $f(x)$ of deg $\leq n-1$. f can be uniquely recovered given its evaluations at $\alpha_1, \alpha_2, \ldots, \alpha_n$.

Pf:

$$f(x) = f_0 + f_1 x + f_2 x^2 + \cdots + f_{n-1} x^{n-1}$$

$$f(\alpha_1) = f_0 + f_1 \alpha_1 + f_2 \alpha_1^2 + \cdots + f_{n-1} \alpha_1^{n-1}$$

$$\begin{cases} f(\alpha_2) = \\ \vdots \\ f(\alpha_n) = f_0 + f_1\alpha_n + \cdots + f_{n-1}\alpha_n^{n-1} \end{cases}$$

$$\underbrace{\begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^{n-1} \\ & & \vdots & & \\ 1 & \alpha_n & \alpha_n^2 & \cdots & \alpha_n^{n-1} \end{bmatrix}}_{\text{Invertible}} \begin{bmatrix} f_0 \\ \vdots \\ f_{n-1} \end{bmatrix} = \begin{bmatrix} f(\alpha_1) \\ f(\alpha_2) \\ \vdots \\ f(\alpha_n) \end{bmatrix}$$

Fact: Det of this matrix $= \prod_{i>j} (\alpha_i - \alpha_j)$

(Vandermonde Matrix)

Solve the system + solution is unique.