

## Divide and Conquer: Matrix Multiplication

CMSC 451

### Matrix Multiplication

**Matrix multiplication.** Given two  $n$ -by- $n$  matrices  $A$  and  $B$ , compute  $C = AB$ .  
**Grade-school.**  $\Theta(n^3)$  arithmetic operations.

$$c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$$

$$\begin{bmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \dots & c_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{bmatrix}$$

$$\begin{bmatrix} .59 & .32 & .41 \\ .31 & .36 & .25 \\ .45 & .31 & .42 \end{bmatrix} = \begin{bmatrix} .70 & .20 & .10 \\ .30 & .60 & .10 \\ .50 & .10 & .40 \end{bmatrix} \times \begin{bmatrix} .80 & .30 & .50 \\ .10 & .40 & .10 \\ .10 & .30 & .40 \end{bmatrix}$$

**Q.** Is grade-school matrix multiplication algorithm optimal?  
 How could we use divide and conquer to multiply faster?

### Block Matrix Multiplication

$$\begin{bmatrix} 152 & 158 & 164 & 170 \\ 504 & 526 & 548 & 570 \\ 856 & 894 & 932 & 970 \\ 1208 & 1262 & 1316 & 1370 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{bmatrix} \times \begin{bmatrix} 16 & 17 & 18 & 19 \\ 20 & 21 & 22 & 23 \\ 24 & 25 & 26 & 27 \\ 28 & 29 & 30 & 31 \end{bmatrix}$$

$$C_{11} = A_{11} \times B_{11} + A_{12} \times B_{21} = \begin{bmatrix} 0 & 1 \\ 4 & 5 \end{bmatrix} \times \begin{bmatrix} 16 & 17 \\ 20 & 21 \end{bmatrix} + \begin{bmatrix} 2 & 3 \\ 6 & 7 \end{bmatrix} \times \begin{bmatrix} 24 & 25 \\ 28 & 29 \end{bmatrix} = \begin{bmatrix} 152 & 158 \\ 504 & 526 \end{bmatrix}$$

### Matrix Multiplication: Warmup

**To multiply two  $n$ -by- $n$  matrices  $A$  and  $B$ :**

- Divide: partition  $A$  and  $B$  into  $\frac{1}{2}n$ -by- $\frac{1}{2}n$  blocks.
- Conquer: multiply 8 pairs of  $\frac{1}{2}n$ -by- $\frac{1}{2}n$  matrices, recursively.
- Combine: add appropriate products using 4 matrix additions.

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$\begin{aligned} C_{11} &= (A_{11} \times B_{11}) + (A_{12} \times B_{21}) \\ C_{12} &= (A_{11} \times B_{12}) + (A_{12} \times B_{22}) \\ C_{21} &= (A_{21} \times B_{11}) + (A_{22} \times B_{21}) \\ C_{22} &= (A_{21} \times B_{12}) + (A_{22} \times B_{22}) \end{aligned}$$

$$T(n) = \underbrace{8T(n/2)}_{\text{recursive calls}} + \underbrace{\Theta(n^2)}_{\text{add, form submatrices}} \Rightarrow T(n) = \Theta(n^3)$$

### Fast Matrix Multiplication

**Key idea.** multiply 2-by-2 blocks with only **7 multiplications**.

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$\begin{aligned} P_1 &= A_{11} \times (B_{12} - B_{22}) \\ P_2 &= (A_{11} + A_{12}) \times B_{22} \\ P_3 &= (A_{21} + A_{22}) \times B_{11} \\ P_4 &= A_{22} \times (B_{21} - B_{11}) \\ P_5 &= (A_{11} + A_{22}) \times (B_{11} + B_{22}) \\ P_6 &= (A_{12} - A_{22}) \times (B_{21} + B_{22}) \\ P_7 &= (A_{11} - A_{21}) \times (B_{11} + B_{12}) \end{aligned}$$

$$\begin{aligned} C_{11} &= P_5 + P_4 - P_2 + P_6 \\ C_{12} &= P_1 + P_2 \\ C_{21} &= P_3 + P_4 \\ C_{22} &= P_3 - P_1 - P_3 - P_7 \end{aligned}$$

- 7 multiplications.
- 18 = 8 + 10 additions and subtractions.

### Fast Matrix Multiplication

**To multiply two  $n$ -by- $n$  matrices  $A$  and  $B$ :** [Strassen 1969]

- Divide: partition  $A$  and  $B$  into  $\frac{1}{2}n$ -by- $\frac{1}{2}n$  blocks.
- Compute: 14  $\frac{1}{2}n$ -by- $\frac{1}{2}n$  matrices via 10 matrix additions.
- Conquer: multiply 7 pairs of  $\frac{1}{2}n$ -by- $\frac{1}{2}n$  matrices, recursively.
- Combine: 7 products into 4 terms using 8 matrix additions.

**Analysis.**

- Assume  $n$  is a power of 2.
- $T(n) = \#$  arithmetic operations.

$$T(n) = \underbrace{7T(n/2)}_{\text{recursive calls}} + \underbrace{\Theta(n^2)}_{\text{add, subtract}} \Rightarrow T(n) = \Theta(n^{\log_2 7}) = O(n^{2.81})$$

Fast Matrix Multiplication: Practice

Implementation issues.

- Sparsity.
- Caching effects.
- Numerical stability.
- Odd matrix dimensions.
- Crossover to classical algorithm around  $n = 128$ .

Common misperception. "Strassen is only a theoretical curiosity."

- Apple reports 8x speedup on G4 Velocity Engine when  $n = 2,500$ .
- Range of instances where it's useful is a subject of controversy.

Remark. Can "Strassenize"  $Ax = b$ , determinant, eigenvalues, SVD, ....

7

Fast Matrix Multiplication: Theory

Q. Multiply two 2-by-2 matrices with 7 scalar multiplications?

A. Yes! [Strassen 1969]  $\Theta(n^{\log_2 7}) = O(n^{2.807})$

Q. Multiply two 2-by-2 matrices with 6 scalar multiplications?

A. Impossible. [Hopcroft and Kerr 1971]  $\Theta(n^{\log_2 6}) = O(n^{2.59})$

Q. Two 3-by-3 matrices with 21 scalar multiplications?

A. Also impossible.  $\Theta(n^{\log_3 21}) = O(n^{2.77})$

Began, the decimal wars have. [Pan, Bini et al, Schönhage, ...]

- Two 20-by-20 matrices with 4,460 scalar multiplications.  $O(n^{2.805})$
- Two 48-by-48 matrices with 47,217 scalar multiplications.  $O(n^{2.7801})$
- A year later.  $O(n^{2.7799})$
- December, 1979.  $O(n^{2.521813})$
- January, 1980.  $O(n^{2.521801})$

8

Fast Matrix Multiplication: Theory

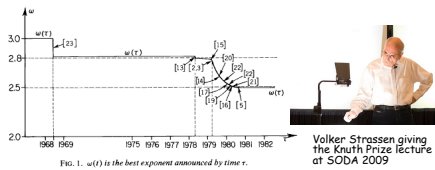


FIG. 1.  $\omega(t)$  is the best exponent announced by time  $t$ .

Best known.  $O(n^{2.376})$  [Coppersmith-Winograd, 1987]

Conjecture.  $O(n^{2+\epsilon})$  for any  $\epsilon > 0$ .

Caveat. Theoretical improvements to Strassen are progressively less practical.

9

Homework Review

Algorithm Write-up Guidelines (see PDF on website)

- 1) Description of Algorithm
- 2) Pseudocode
- 3) Time Analysis
- 4) Proof of correctness (the algorithm does what you say it does and is optimal)

Group work - put all names on the paper, hand in a single copy!

10