

CSCI 355: ALGORITHM DESIGN AND ANALYSIS 6. DIVIDE AND CONQUER II

- ▶ master theorem
- ▶ integer multiplication
- ▶ matrix multiplication

Divide-and-conquer recurrences

Goal. A recipe for solving common divide-and-conquer recurrences:

$$T(n) = a T\left(\frac{n}{b}\right) + f(n)$$

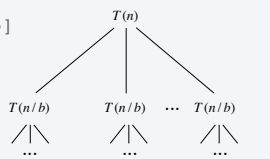
with $T(0) = 0$ and $T(1) = \Theta(1)$.

Terms.

- $a \geq 1$ is the number of subproblems.
- $b \geq 2$ is the factor by which the subproblem size decreases.
- $f(n) \geq 0$ is the work to divide and combine subproblems.

Recursion tree. [assuming n is a power of b]

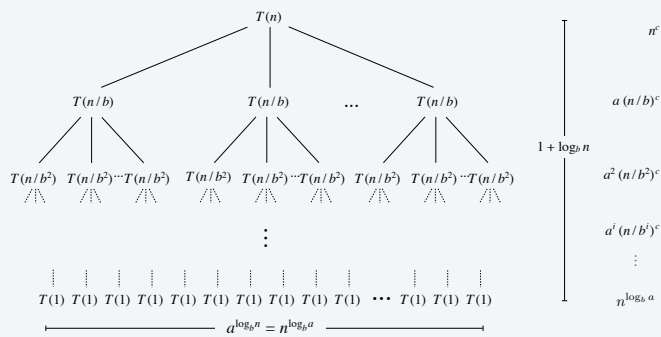
- a = branching factor.
- a^i = number of subproblems at level i .
- $1 + \log_b n$ levels.
- n / b^i = size of subproblem at level i .



3

Divide-and-conquer recurrences: recursion tree

Suppose $T(n)$ satisfies $T(n) = a T(n/b) + n^c$ with $T(1) = 1$, for n a power of b .



$$r = a / b^c \quad T(n) = n^c \sum_{i=0}^{\log_b n} r^i$$

4

Divide-and-conquer recurrences: recursion tree

Suppose $T(n)$ satisfies $T(n) = aT(n/b) + n^c$ with $T(1) = 1$, for n a power of b .

Let $r = a/b^c$. Note that $r < 1$ if and only if $c > \log_b a$.

$$T(n) = n^c \sum_{i=0}^{\log_b n} r^i = \begin{cases} \Theta(n^c) & \text{if } r < 1 \quad c > \log_b a \quad \leftarrow \text{cost dominated by cost of root} \\ \Theta(n^c \log n) & \text{if } r = 1 \quad c = \log_b a \quad \leftarrow \text{cost evenly distributed in tree} \\ \Theta(n^{\log_b a}) & \text{if } r > 1 \quad c < \log_b a \quad \leftarrow \text{cost dominated by cost of leaves} \end{cases}$$

Geometric series.

- If $0 < r < 1$, then $1 + r + r^2 + r^3 + \dots + r^k \leq 1 / (1 - r)$.
- If $r = 1$, then $1 + r + r^2 + r^3 + \dots + r^k = k + 1$.
- If $r > 1$, then $1 + r + r^2 + r^3 + \dots + r^k = (r^{k+1} - 1) / (r - 1)$.

5

Divide-and-conquer recurrences: master theorem

Master theorem. Let $a \geq 1$, $b \geq 2$, and $c > 0$ and suppose that $T(n)$ is a function on the non-negative integers that satisfies the recurrence

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^c)$$

with $T(0) = 0$ and $T(1) = \Theta(1)$, where n/b means either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then,

Case 1. If $c < \log_b a$, then $T(n) = \Theta(n^{\log_b a})$.

Case 2. If $c = \log_b a$, then $T(n) = \Theta(n^c \log n)$.

Case 3. If $c > \log_b a$, then $T(n) = \Theta(n^c)$.



Bentley Haken Saxe

Proof sketch.

- Prove when b is an integer and n is an exact power of b .
- Extend domain of recurrences to reals (or rationals).
- Deal with floors and ceilings. \leftarrow at most 2 extra levels in recursion tree

$$\begin{aligned} \lceil \lceil n/b \rceil / b \rceil &< n/b^3 + (1/b^2 + 1/b + 1) \\ &\leq n/b^3 + 2 \end{aligned}$$

6

Divide-and-conquer recurrences: master theorem

Master theorem. Let $a \geq 1$, $b \geq 2$, and $c > 0$ and suppose that $T(n)$ is a function on the non-negative integers that satisfies the recurrence

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^c)$$

with $T(0) = 0$ and $T(1) = \Theta(1)$, where n/b means either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then,

Case 1. If $c < \log_b a$, then $T(n) = \Theta(n^{\log_b a})$.

Case 2. If $c = \log_b a$, then $T(n) = \Theta(n^c \log n)$.

Case 3. If $c > \log_b a$, then $T(n) = \Theta(n^c)$.



Bentley Haken Saxe

Extensions.

- Can replace Θ with O everywhere.
- Can replace Θ with Ω everywhere.
- Can replace initial conditions with $T(n) = \Theta(1)$ for all $n \leq n_0$ and require the recurrence to hold only for all $n > n_0$.

7

Divide-and-conquer recurrences: master theorem

Master theorem. Let $a \geq 1$, $b \geq 2$, and $c > 0$ and suppose that $T(n)$ is a function on the non-negative integers that satisfies the recurrence

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^c)$$

with $T(0) = 0$ and $T(1) = \Theta(1)$, where n/b means either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then,

Case 1. If $c < \log_b a$, then $T(n) = \Theta(n^{\log_b a})$.

Case 2. If $c = \log_b a$, then $T(n) = \Theta(n^c \log n)$.

Case 3. If $c > \log_b a$, then $T(n) = \Theta(n^c)$.



Ex 1. $T(n) = 3T(\lfloor n/2 \rfloor) + 5n$.

- $a = 3$, $b = 2$, $c = 1$
- $\log_b a = 1.58 (> c)$.
- $T(n) = \Theta(n^{\log_2 3}) = \Theta(n^{1.58})$.

8

Divide-and-conquer recurrences: master theorem

Master theorem. Let $a \geq 1$, $b \geq 2$, and $c > 0$ and suppose that $T(n)$ is a function on the non-negative integers that satisfies the recurrence

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^c)$$

with $T(0) = 0$ and $T(1) = \Theta(1)$, where n/b means either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then,

Case 1. If $c < \log_b a$, then $T(n) = \Theta(n^{\log_b a})$.

Case 2. If $c = \log_b a$, then $T(n) = \Theta(n^c \log n)$.

Case 3. If $c > \log_b a$, then $T(n) = \Theta(n^c)$.



okay to intermix floor and ceiling

Ex 2. $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 17n$.

- $a = 2$, $b = 2$, $c = 1$
- $\log_b a = 1 (= c)$
- $T(n) = \Theta(n \log n)$.

9

Divide-and-conquer recurrences: master theorem

Master theorem. Let $a \geq 1$, $b \geq 2$, and $c > 0$ and suppose that $T(n)$ is a function on the non-negative integers that satisfies the recurrence

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^c)$$

with $T(0) = 0$ and $T(1) = \Theta(1)$, where n/b means either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then,

Case 1. If $c < \log_b a$, then $T(n) = \Theta(n^{\log_b a})$.

Case 2. If $c = \log_b a$, then $T(n) = \Theta(n^c \log n)$.

Case 3. If $c > \log_b a$, then $T(n) = \Theta(n^c)$.



Ex 3. $T(n) = 48T(\lfloor n/4 \rfloor) + n^3$.

- $a = 48$, $b = 4$, $c = 3$,
- $\log_b a = 2.79 (< c)$
- $T(n) = \Theta(n^3)$.

10

Divide-and-conquer recurrences: master theorem

Gaps in the master theorem (inadmissible equations).

- Number of subproblems is not a constant.

$$T(n) = \Theta T(n/2) + n^2$$

- Number of subproblems is less than 1.

$$T(n) = \left(\frac{1}{2}\right) T(n/2) + n^2$$

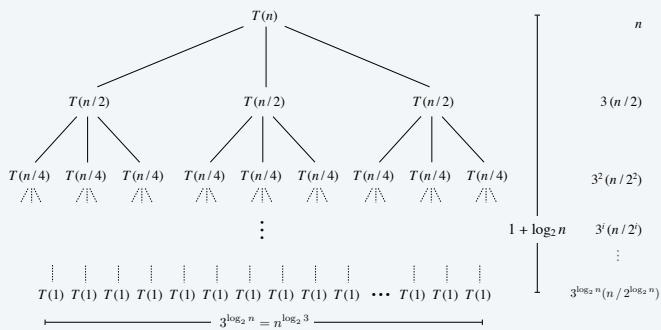
- Work done to divide and combine subproblems is not $\Theta(n^c)$.

$$T(n) = 2T(n/2) + n \log n$$

11

Recurrence tree: cost dominated by cost of leaves

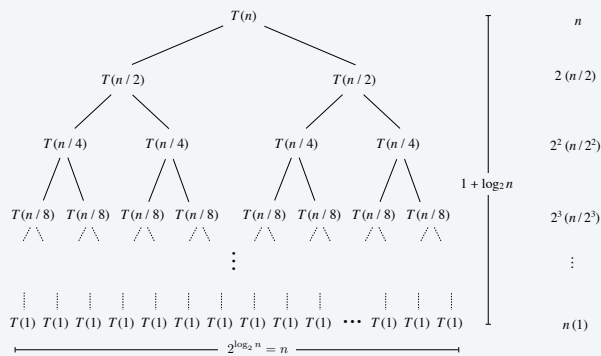
Ex 1. If $T(n)$ satisfies $T(n) = 3T(n/2) + n$, with $T(1) = 1$, then $T(n) = \Theta(n^{\log_2 3})$.



13

Recurrence tree: cost evenly distributed among levels

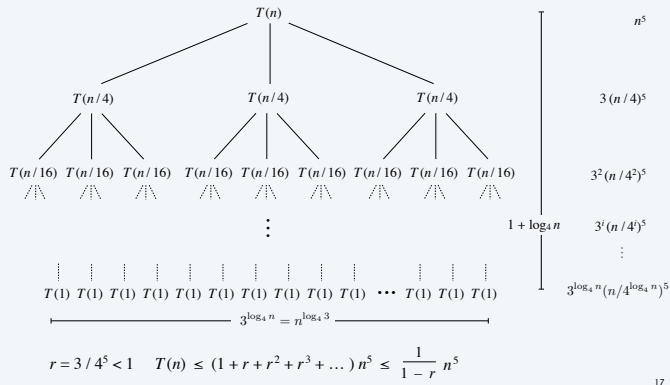
Ex 2. If $T(n)$ satisfies $T(n) = 2T(n/2) + n$, with $T(1) = 1$, then $T(n) = \Theta(n \log n)$.



15

Recurrence tree: cost dominated by cost of root

Ex 3. If $T(n)$ satisfies $T(n) = 3T(n/4) + n^5$, with $T(1) = 1$, then $T(n) = \Theta(n^5)$.



CSCI 355: ALGORITHM DESIGN AND ANALYSIS
6. DIVIDE AND CONQUER II

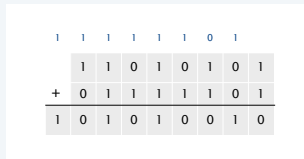
- ▶ master theorem
- ▶ integer multiplication
- ▶ matrix multiplication

Integer addition and subtraction

Addition. Given two n -bit integers a and b , compute $a + b$.

Subtraction. Given two n -bit integers a and b , compute $a - b$.

Grade school algorithm. $\Theta(n)$ bit operations. ← "bit complexity" (instead of word RAM)

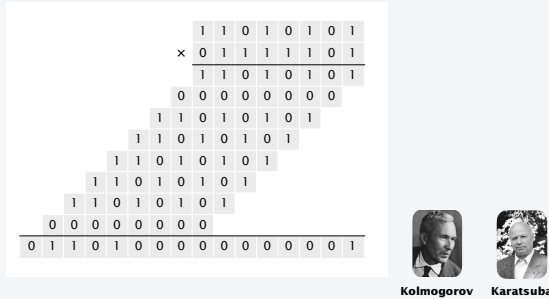


Remark. Grade school addition and subtraction algorithms are optimal.

Integer multiplication

Multiplication. Given two n -bit integers a and b , compute $a \times b$.

Grade school algorithm (long multiplication). $\Theta(n^2)$ bit operations.



Conjecture. [Kolmogorov 1956] Grade-school algorithm is optimal.

Theorem. [Karatsuba 1960] Conjecture is false.

20

Integer multiplication: divide-and-conquer

To multiply two n -bit integers x and y :

- Divide x and y into low- and high-order bits.
- Multiply *four* $\frac{1}{2}n$ -bit integers, recursively.
- Add and shift to obtain result.

$$m = \lceil n / 2 \rceil$$

$$a = \lfloor x / 2^m \rfloor \quad b = x \bmod 2^m$$

$$c = \lfloor y / 2^m \rfloor \quad d = y \bmod 2^m$$

← use bit shifting to compute 4 terms

$$x y = (2^m a + b)(2^m c + d) = 2^{2m} ac + 2^m (bc + ad) + bd$$

① ② ③ ④

Ex. $x = \underbrace{1000}_a \underbrace{1101}_b \quad y = \underbrace{1110}_c \underbrace{0001}_d$

21

Integer multiplication: divide-and-conquer

MULTIPLY(x, y, n)

IF ($n = 1$)

RETURN xy .

ELSE

$m \leftarrow \lceil n / 2 \rceil$.

$a \leftarrow \lfloor x / 2^m \rfloor$; $b \leftarrow x \bmod 2^m$.

$c \leftarrow \lfloor y / 2^m \rfloor$; $d \leftarrow y \bmod 2^m$.

$e \leftarrow \text{MULTIPLY}(a, c, m)$.

$f \leftarrow \text{MULTIPLY}(b, d, m)$.

$g \leftarrow \text{MULTIPLY}(b, c, m)$.

$h \leftarrow \text{MULTIPLY}(a, d, m)$.

RETURN $2^{2m} e + 2^m (g + h) + f$.

← $\Theta(n)$

← $4T(n/2)$

← $\Theta(n)$

22

Karatsuba's trick

To multiply two n -bit integers x and y :

- Divide x and y into low- and high-order bits.
- To compute the middle term $bc + ad$, use the identity:

$$bc + ad = ac + bd - (a - b)(c - d)$$

- Multiply only *three* $\frac{1}{2}n$ -bit integers, recursively.

$$\begin{aligned}
 m &= \lceil n / 2 \rceil \\
 a &= \lfloor x / 2^m \rfloor \quad b = x \bmod 2^m \\
 c &= \lfloor y / 2^m \rfloor \quad d = y \bmod 2^m \\
 xy &= (2^m a + b)(2^m c + d) = 2^{2m} ac + 2^m (bc + ad) + bd \\
 &= 2^{2m} ac + 2^m (ac + bd - (a - b)(c - d)) + bd
 \end{aligned}$$

24

Integer multiplication: Karatsuba's algorithm

KARATSUBA-MULTIPLY(x, y, n)

IF ($n = 1$)

 RETURN xy .

ELSE

$m \leftarrow \lceil n / 2 \rceil$.

$a \leftarrow \lfloor x / 2^m \rfloor$; $b \leftarrow x \bmod 2^m$. $\leftarrow \Theta(n)$

$c \leftarrow \lfloor y / 2^m \rfloor$; $d \leftarrow y \bmod 2^m$.

$e \leftarrow$ KARATSUBA-MULTIPLY(a, c, m).

$f \leftarrow$ KARATSUBA-MULTIPLY(b, d, m).

$g \leftarrow$ KARATSUBA-MULTIPLY($|a - b|, |c - d|, m$). $\leftarrow 3 T(n/2)$

 Flip sign of g if needed.

 RETURN $2^{2m} e + 2^m (e + f - g) + f$. $\leftarrow \Theta(n)$

25

Karatsuba's algorithm: analysis

Proposition. Karatsuba's algorithm requires $O(n^{1.585})$ bit operations to multiply two n -bit integers.

Pf. Apply Case 1 of the master theorem to the recurrence:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 3T(\lceil n/2 \rceil) + \Theta(n) & \text{if } n > 1 \end{cases}$$

$$\implies T(n) = \Theta(n^{\log_2 3}) = O(n^{1.585})$$

In practice.

- Use base 32 or 64 (instead of base 2).
- Faster than grade-school algorithm for about 320–640 bits.

26

Integer arithmetic reductions

arithmetic problem	formula	bit complexity
integer multiplication	$a \times b$	$M(n)$
integer squaring	a^2	$\Theta(M(n))$
integer division	$\lfloor a / b \rfloor, a \bmod b$	$\Theta(M(n))$
integer square root	$\lfloor \sqrt{a} \rfloor$	$\Theta(M(n))$

$$ab = \frac{(a+b)^2 - a^2 - b^2}{2}$$

integer arithmetic problems with the same bit complexity $M(n)$ as integer multiplication

27

Integer multiplication in loglinear time

Integer multiplication. Given two n -bit integers $a = a_{n-1} \dots a_1 a_0$ and $b = b_{n-1} \dots b_1 b_0$, compute their product $a \cdot b$.

Convolution algorithm.

- Form two polynomials. $A(x) = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1}$
- Note: $a = A(2), b = B(2)$. $B(x) = b_0 + b_1x + b_2x^2 + \dots + b_{n-1}x^{n-1}$
- Compute $C(x) = A(x) \cdot B(x)$.
- Evaluate $C(2) = a \cdot b$.
- Running time: $O(n \log n)$ floating-point operations.

Analysis. [Schönhage–Strassen 1971]

- $O(n \log^2 n)$ bit operations. ← FFT over complex numbers; need $O(\log n)$ bits of precision
- $O(n \log n \cdot \log \log n)$ bit operations. ← FFT over ring of integers (modulo a Fermat number)

28

History of integer multiplication

year	algorithm	bit operations
antiquity	grade school	$O(n^2)$
1962	Karatsuba–Ofman	$O(n^{1.585})$
1963	Toom–3, Toom–4	$O(n^{1.465}), O(n^{1.404})$
1966	Toom–Cook	$O(n^{1+\epsilon})$
1971	Schönhage–Strassen	$O(n \log n \cdot \log \log n)$
2007	Fürer	$n \log n 2^{O(\log^3 n)}$
2019	Harvey–van der Hoeven	$O(n \log n)$
	???	$O(n)$

number of bit operations to multiply two n -bit integers

Remark. GNU Multiple Precision Library uses one of the first five algorithms depending on n .

used in Maple, Mathematica, gcc, cryptography, ...

GMP
«Arithmetic without limitations»

29

CSCI 355: ALGORITHM DESIGN AND ANALYSIS

6. DIVIDE AND CONQUER II

- ▶ *master theorem*
- ▶ *integer multiplication*
- ▶ *matrix multiplication*

Dot product

Dot product. Given two length- n vectors a and b , compute $c = a \cdot b$.

Grade school algorithm. $\Theta(n)$ arithmetic operations.

$$a \cdot b = \sum_{i=1}^n a_i b_i$$

$$\begin{aligned} a &= [.70 \quad .20 \quad .10] \\ b &= [.30 \quad .40 \quad .30] \\ a \cdot b &= (.70 \times .30) + (.20 \times .40) + (.10 \times .30) = .32 \end{aligned}$$

Remark. Grade school dot product algorithm is asymptotically optimal.

31

Matrix multiplication

Matrix multiplication. Given two n -by- n matrices A and B , compute $C = AB$.

Grade school algorithm. $\Theta(n^3)$ 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 asymptotically optimal?

32

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} points to the top-left 2x2 block of the result matrix.
 A_{11} and A_{12} point to the top two rows of matrix A.
 B_{11} and B_{21} point to the first two columns of matrix B.

33

Block matrix multiplication

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.

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

n -by- n matrices (pointing to A and B)
 $\frac{1}{2}n$ -by- $\frac{1}{2}n$ matrices (pointing to sub-blocks of A and B)
 8 matrix multiplications (of $\frac{1}{2}n$ -by- $\frac{1}{2}n$ matrices) (pointing to the 8 products in the equations)
 4 matrix additions (of $\frac{1}{2}n$ -by- $\frac{1}{2}n$ matrices) (pointing to the 4 additions in the equations)

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

Running time. Apply Case 1 of the master theorem.

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

34

Strassen's trick

Key idea. We can multiply two 2-by-2 matrices via 7 scalar multiplications (plus 11 additions and 7 subtractions).

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

scalars (pointing to the sub-blocks of A and B)

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

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

7 scalar multiplications (pointing to the P equations)

Pf. $C_{12} = P_1 + P_2$

$$\begin{aligned}
 &= A_{11} \times (B_{12} - B_{22}) + (A_{11} + A_{12}) \times B_{22} \\
 &= A_{11} \times B_{12} + A_{12} \times B_{22}. \quad \checkmark
 \end{aligned}$$

35

Strassen's trick

Key idea. We can multiply two n -by- n matrices via 7 scalar multiplications (plus 11 additions and 7 subtractions).

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

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

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

Pf. $C_{12} = P_1 + P_2$
 $= A_{11} \times (B_{12} - B_{22}) + (A_{11} + A_{12}) \times B_{22}$
 $= A_{11} \times B_{12} + A_{12} \times B_{22}. \quad \checkmark$

7 matrix multiplications
(of $\frac{1}{2}n$ -by- $\frac{1}{2}n$ matrices)

36

Block matrix multiplication: Strassen's algorithm

STRASSEN(n, A, B)

IF ($n = 1$) RETURN $A \times B$.

Partition A and B into $\frac{1}{2}n$ -by- $\frac{1}{2}n$ blocks.

$P_1 \leftarrow$ STRASSEN($n/2, A_{11}, (B_{12} - B_{22})$).

$P_2 \leftarrow$ STRASSEN($n/2, (A_{11} + A_{12}), B_{22}$).

$P_3 \leftarrow$ STRASSEN($n/2, (A_{21} + A_{22}), B_{11}$).

$P_4 \leftarrow$ STRASSEN($n/2, A_{22}, (B_{21} - B_{11})$).

$P_5 \leftarrow$ STRASSEN($n/2, (A_{11} + A_{22}), (B_{11} + B_{22})$).

$P_6 \leftarrow$ STRASSEN($n/2, (A_{12} - A_{22}), (B_{21} + B_{22})$).

$P_7 \leftarrow$ STRASSEN($n/2, (A_{11} - A_{21}), (B_{11} + B_{12})$).

$C_{11} = P_5 + P_4 - P_2 + P_6$.

$C_{12} = P_1 + P_2$.

$C_{21} = P_3 + P_4$.

$C_{22} = P_1 + P_5 - P_3 - P_7$.

RETURN C .

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

$7T(n/2) + \Theta(n^2)$

$\Theta(n^2)$

assume n is a power of 2

37

Strassen's algorithm: analysis

Theorem. Strassen's algorithm requires $O(n^{2.81})$ arithmetic operations to multiply two n -by- n matrices.

Gaussian Elimination is not Optimal

VOLKER STRASSEN*

Received December 12, 1968

1. Below we will give an algorithm which computes the coefficients of the product of two square matrices A and B of order n from the coefficients of A and B with less than $4.7 \cdot n^{2.81}$ arithmetical operations (all logarithms in this paper are for base 2, thus $\log 7 \approx 2.8$; the usual method requires approximately $2n^3$ arithmetical operations). The algorithm induces algorithms for inverting a matrix of order n , solving a system of n linear equations in n unknowns, computing a determinant of order n etc. all requiring less than const $n^{2.81}$ arithmetical operations.



Strassen

38

Strassen's algorithm: analysis

Theorem. Strassen's algorithm requires $O(n^{2.81})$ arithmetic operations to multiply two n -by- n matrices.

Pf.

- When n is a power of 2, apply Case 1 of the master theorem:

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

- When n is not a power of 2, pad matrices with zeroes to be n' -by- n' , where $n \leq n' < 2n$ and n' is a power of 2.

$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 4 & 5 & 6 & 0 \\ 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 10 & 11 & 12 & 0 \\ 13 & 14 & 15 & 0 \\ 16 & 17 & 18 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 84 & 90 & 96 & 0 \\ 201 & 216 & 231 & 0 \\ 318 & 342 & 366 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

39

Strassen's algorithm: practice

Implementation issues.

- Sparsity.
- Caching.
- n may not be a power of 2.
- Numerical stability.
- Non-square matrices.
- Storage for intermediate submatrices.
- Crossover to the classical algorithm when n is "small."
- Parallelism for multi-core and many-core architectures.

Common misperception. "Strassen's algorithm is only a theoretical curiosity."

- Research has reported an 8x speedup when $n \approx 2,048$.
- Range of instances where it's useful is a subject of controversy.

Strassen's Algorithm Reloaded

Gregory W. Moore, David M. Blewett, Greg M. Blewett, Robert A. van de Geijl
 Department of Computer Science and Research for Intelligent Engineering and Systems,
 The University of Queensland, St. Lucia, QLD, 4072
 Email: greg@cs.uq.edu.au
 Greg.M.Blewett@uq.edu.au
 Email: greg.blewett@uq.edu.au

40

Numeric linear algebra reductions

linear algebra problem	expression	arithmetic complexity
matrix multiplication	$A \times B$	$MM(n)$
matrix squaring	A^2	$\Theta(MM(n))$
matrix inversion	A^{-1}	$\Theta(MM(n))$
determinant	$ A $	$\Theta(MM(n))$
rank	$\text{rank}(A)$	$\Theta(MM(n))$
system of linear equations	$Ax = b$	$\Theta(MM(n))$
LU decomposition	$A = LU$	$\Theta(MM(n))$
least squares	$\min \ Ax - b\ _2$	$\Theta(MM(n))$

numerical linear algebra problems with the same arithmetic complexity $MM(n)$ as matrix multiplication

41

Fast matrix multiplication

Q. Can we multiply two 2-by-2 matrices with 7 scalar multiplications?

A. Yes! [Strassen 1969] $\Theta(n^{\log_2 7}) = O(n^{2.81})$

Q. Can we multiply two 2-by-2 matrices with 6 scalar multiplications?

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

Race to n^2 . [Pan 1978, Bini et al. 1979, Schönhage 1981, ...]

- Two 70-by-70 matrices with 143,640 scalar multiplications. $O(n^{2.7962})$
- Two 48-by-48 matrices with 47,217 scalar multiplications. $O(n^{2.7801})$

42

History of matrix multiplication

year	algorithm	arithmetic operations
1858	grade school	$O(n^3)$
1969	Strassen	$O(n^{2.81})$
1978	Pan	$O(n^{2.796})$
1979	Bini-Capovani-Romani	$O(n^{2.780})$
1981	Schönhage	$O(n^{2.522})$
1982	Romani	$O(n^{2.517})$
1982	Coppersmith-Winograd	$O(n^{2.496})$
1986	Strassen	$O(n^{2.479})$
1989	Coppersmith-Winograd	$O(n^{2.3755})$
2010	Stothers	$O(n^{2.3737})$
2011	Williams	$O(n^{2.3729})$
2014	Le Gall	$O(n^{2.3728639})$
2020	Alman-Williams	$O(n^{2.3728596})$
2022*	Duan-Wu-Zhou	$O(n^{2.371866})$
2023*	Williams-Xu-Xu-Zhou	$O(n^{2.371552})$
	???	$O(n^{2+\epsilon})$

galactic algorithms

†
not yet peer-reviewed

number of arithmetic operations to multiply two n -by- n matrices

43