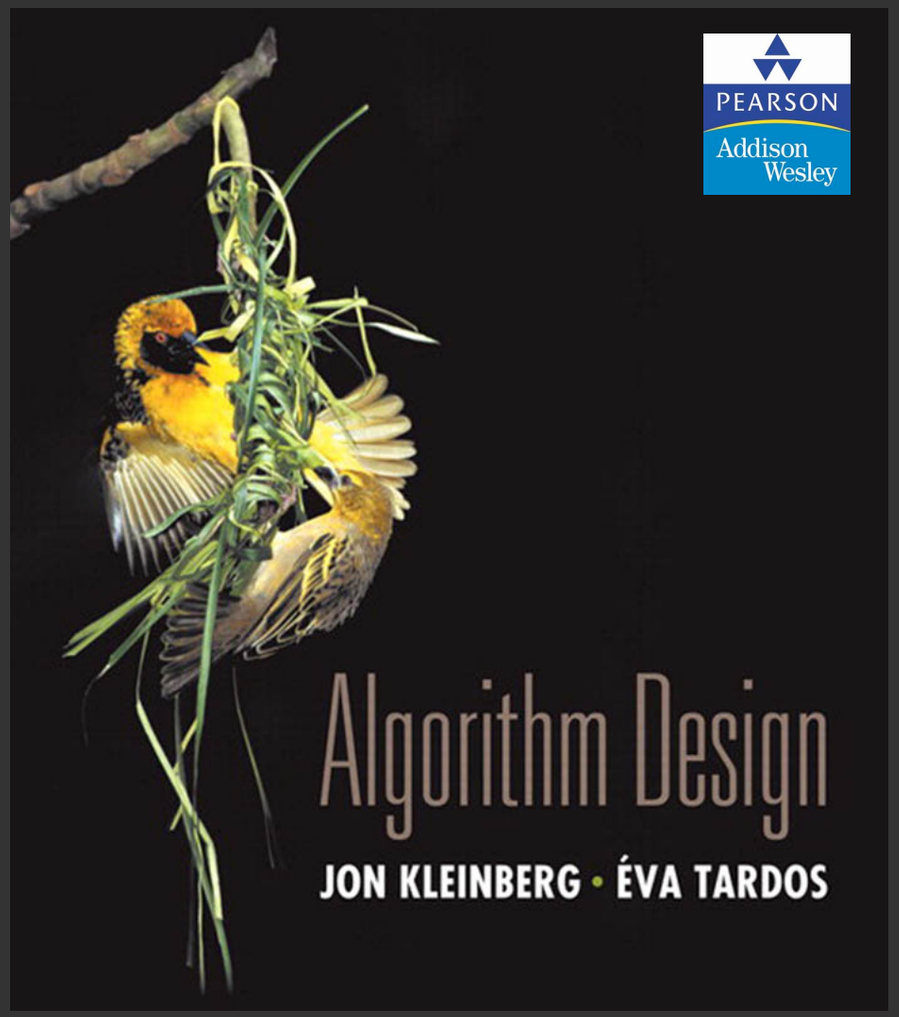
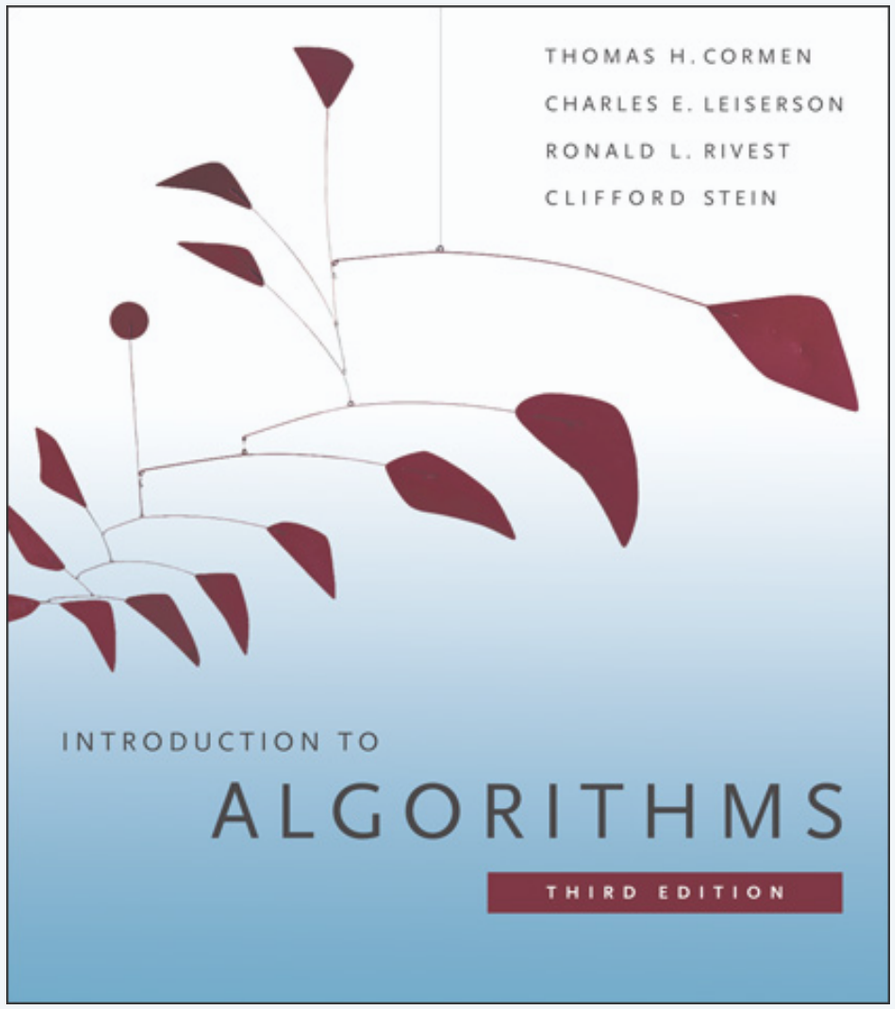


Divide and Conquer II

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

Lecture slides by Kevin Wayne

http://www.cs.princeton.edu/~wayne/kleinberg-tardos



Divide and Conquer II

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

Sections 4.4–4.6

Divide-and-conquer recurrences

Goal. Recipe for solving common divide-and-conquer recurrences:

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

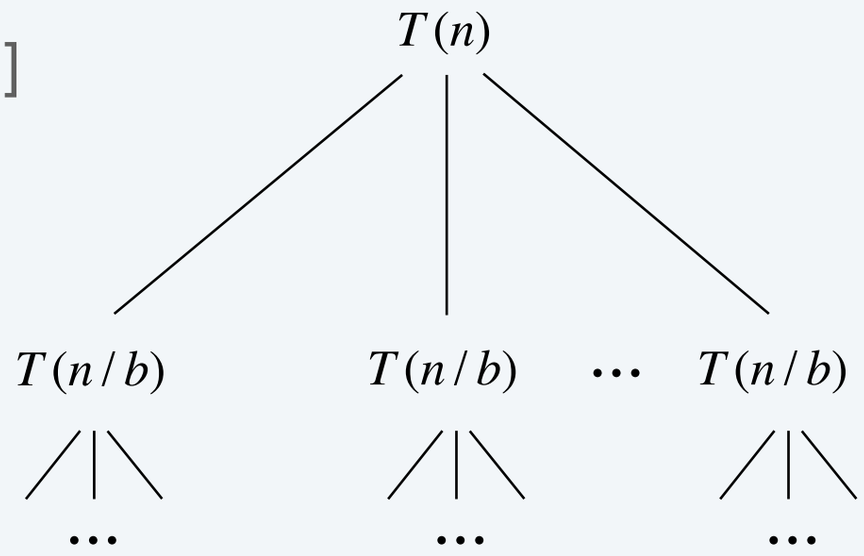
with $T(0) = 0$ and $T(1) = \Theta(1)$.

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.

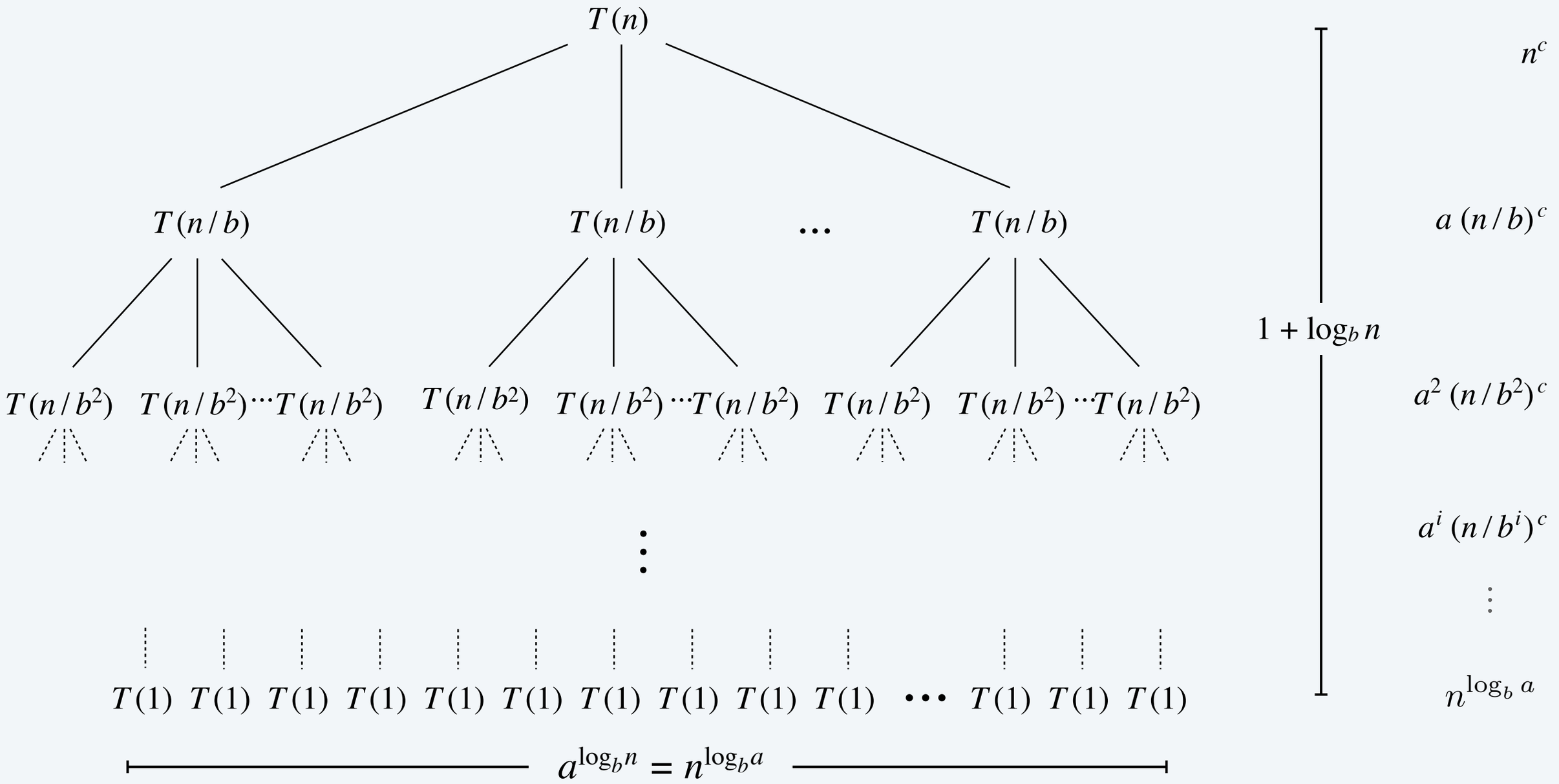
Recursion tree. [assuming n is a power of b]

- 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.



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.



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

Divide-and-conquer recurrences: recursion tree analysis

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

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

$$T(n) \;=\; n^c \sum_{i=0}^{\log_b n} r^i \;=\; \begin{cases} \Theta(n^c) & \text{if } r < 1 \\ \Theta(n^c \log n) & \text{if } r = 1 \\ \Theta(n^{\log_b a}) & \text{if } r > 1 \end{cases}$$

$c > \log_b a$ ← cost dominated by cost of root

$c = \log_b a$ ← cost evenly distributed in tree

$c < \log_b a$ ← cost dominated by cost of leaves

Geometric series.

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

Divide-and-conquer recurrences: master theorem

Master theorem. Let $a \geq 1$, $b \geq 2$, and $c \geq 0$ 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 $T(n) = \Theta(n^c)$.

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

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

Pf 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. ← at most 2 extra levels in recursion tree

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

Divide-and-conquer recurrences: master theorem

Master theorem. Let $a \geq 1$, $b \geq 2$, and $c \geq 0$ 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 $T(n) = \Theta(n^c)$.

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

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

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 recurrence to hold only for all $n > n_0$.

Divide-and-conquer recurrences: master theorem

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

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

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 $T(n) = \Theta(n^c)$.

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

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

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

- $a = 3,\ b = 2,\ c = 1 < \log_b a = 1.5849\ldots$.
- $T(n) = \Theta(n^{\log_2 3}) = \mathrm{O}(n^{1.58})$.

Divide-and-conquer recurrences: master theorem

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

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

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 $T(n) = \Theta(n^c)$.

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

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

ok to intermix floor and ceiling

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

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

Divide-and-conquer recurrences: master theorem

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

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

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 $T(n) = \Theta(n^c)$.

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

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

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

- $a = 48,\ b = 4,\ c = 3 > \log_b a = 2.7924....$
- $T(n) = \Theta(n^3)$.

Master theorem need not apply

Gaps in master theorem.

- Number of subproblems is not a constant.

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

- Number of subproblems is less than 1.

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

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

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

Divide-and-conquer II: quiz 1

Consider the following recurrence. Which case of the master theorem?

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

A. Case 3: $T(n) = \Theta(n)$.

B. Case 2: $T(n) = \Theta(n \log n)$.

C. Case 1: $T(n) = \Theta(n^{\log_2 3}) = O(n^{1.585})$.

D. Master theorem not applicable.

Divide-and-conquer II: quiz 2

Consider the following recurrence. Which case of the master theorem?

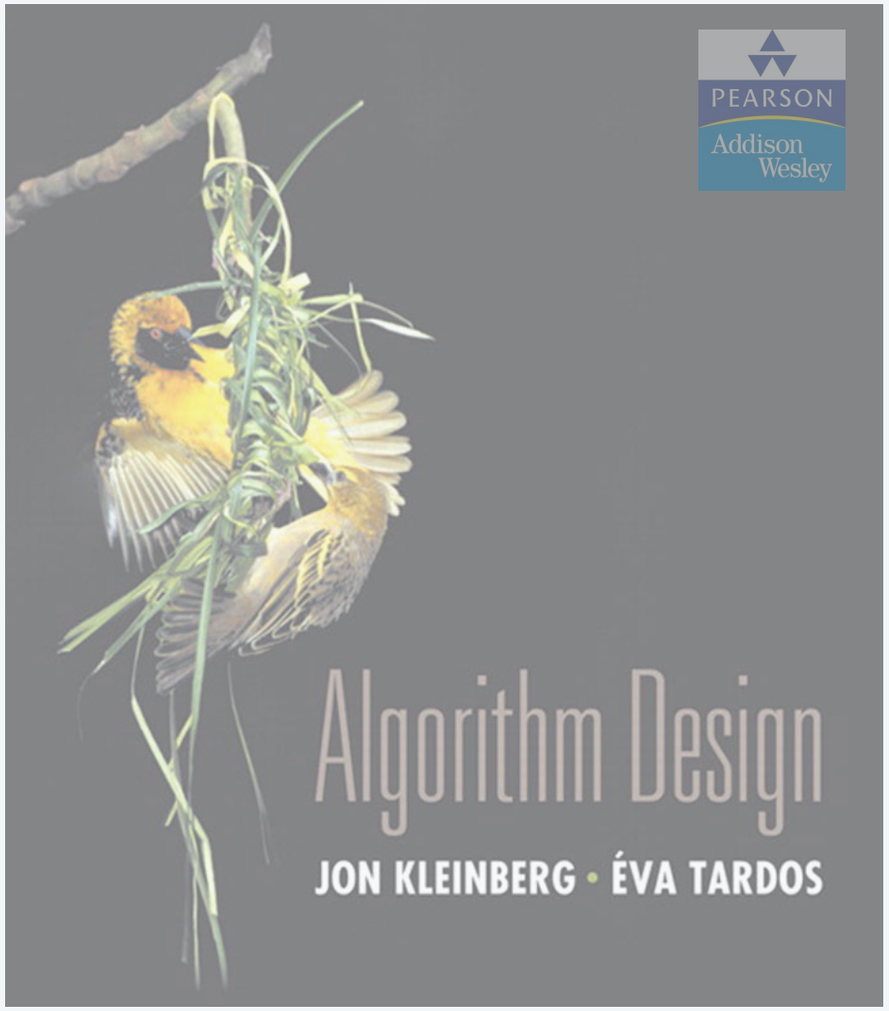
$$T(n) = \begin{cases} 0 & \text{if } n \leq 1 \\ T(\lfloor n/5 \rfloor) + T(n - 3\lfloor n/10 \rfloor) + \frac{11}{5}n & \text{if } n > 1 \end{cases}$$

A. Case 1: $T(n) = \Theta(n)$.

B. Case 2: $T(n) = \Theta(n \log n)$.

C. Case 3: $T(n) = \Theta(n)$.

D. Master theorem not applicable.



DIVIDE AND CONQUER II

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

SECTION 5.5

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)

	1	1	1	1	1	1	0	1
	1	1	0	1	0	1	0	1
+	0	1	1	1	1	1	0	1
1	0	1	0	1	0	0	1	0

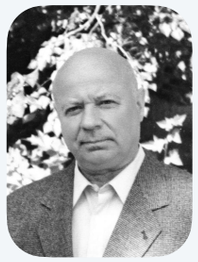
Remark. Grade-school addition and subtraction algorithms are optimal.

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.

	1	1	0	1	0	1	0	1
×	0	1	1	1	1	1	0	1
	1	1	0	1	0	1	0	1
0	0	0	0	0	0	0	0	
1	1	0	1	0	1	0	1	
1	1	0	1	0	1	0	1	
1	1	0	1	0	1	0	1	
1	1	0	1	0	1	0	1	
0	0	0	0	0	0	0	0	
0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1



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

Theorem. [Karatsuba 1960] Conjecture is false.

Divide-and-conquer multiplication

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

$$x\,y = (2^m a + b)(2^m c + d) = \underset{1}{2^{2m} ac} + \underset{2}{2^m (bc} + \underset{3}{ad)} + \underset{4}{bd}$$

Ex. $x = \underbrace{1\,0\,0\,0}_{a}\,\underbrace{1\,1\,0\,1}_{b}$ $\quad y = \underbrace{1\,1\,1\,0}_{c}\,\underbrace{0\,0\,0\,1}_{d}$

Divide-and-conquer multiplication

MULTIPLY(x, y, n)

IF $(n = 1)$

  RETURN $x \times y$.

ELSE

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

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

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

  $\leftarrow \Theta(n)$

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

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

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

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

  $\leftarrow 4\ T(\lceil n / 2 \rceil)$

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

Divide-and-conquer II: quiz 3

How many bit operations to multiply two n-bit integers using the divide-and-conquer multiplication algorithm?

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 4T(\lceil n/2 \rceil) + \Theta(n) & \text{if } n > 1 \end{cases}$$

A. $T(n) = \Theta(n^{1/2})$.

B. $T(n) = \Theta(n \log n)$.

C. $T(n) = \Theta(n^{\log_2 3}) = \mathrm{O}(n^{1.585})$.

D. $T(n) = \Theta(n^2)$.

Karatsuba trick

To multiply two n-bit integers x and y:

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

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

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

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

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

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

$$x = 1\underbrace{0\,0\,0}_{a}\underbrace{0\,1\,1\,0\,1}_{b}$$

$$y = 1\underbrace{1\,1\,0}_{c}\underbrace{0\,0\,0\,1}_{d}$$

middle term

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

$$= 2^{2m} ac + 2^m (ac + bd - (a - b)(c - d)) + bd$$

1 1 3 2 3

Karatsuba multiplication

KARATSUBA-MULTIPLY(x, y, n)

IF $(n = 1)$

  RETURN $x \times y$.

ELSE

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

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

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

  $\leftarrow \Theta(n)$

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

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

  $g \leftarrow$ KARATSUBA-MULTIPLY$(|a - b|, |c - d|, m)$.

  $\leftarrow 3\ T(\lceil n / 2 \rceil)$

  Flip sign of g if needed.

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

Karatsuba analysis

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

Pf. Apply Case 3 of the master theorem to the recurrence:

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

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

Practice.

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

Integer arithmetic reductions

Integer multiplication. Given two n-bit integers, compute their product.

arithmetic problem	formula	bit complexity
integer multiplication	$a \times b$	$M(n)$
integer square	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

History of asymptotic complexity of integer multiplication

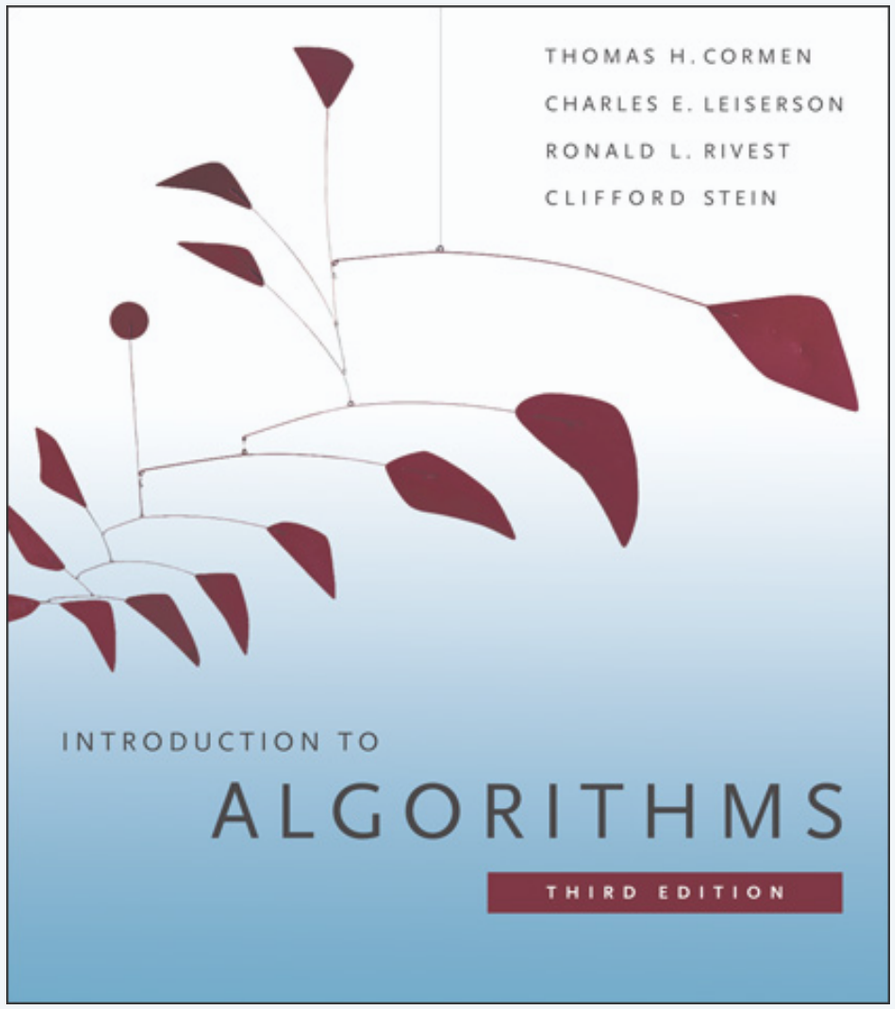
year	algorithm	bit operations
12xx	**grade school**	$O(n^2)$
1962	**Karatsuba–Ofman**	$O(n^{1.585})$
1963	**Toom-3, Toom-4**	$O(n^{1.465})$, $O(n^{1.404})$
1966	**Toom–Cook**	$O(n^{1+\varepsilon})$
1971	**Schönhage–Strassen**	$O(n \log n \cdot \log \log n)$
2007	**Fürer**	$n \log n \, 2^{O(\log^* n)}$
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 first five algorithms depending on n.

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

GMP
«Arithmetic without limitations»



SECTION 4.2

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. $\Theta(n)$ arithmetic operations.

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

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

Remark. "Grade-school" dot product algorithm is asymptotically optimal.

Matrix multiplication

Matrix multiplication. Given two n-by-n matrices A and B, compute $\underbrace{C = AB}_{n}$.

Grade-school. $\Theta(n^3)$ arithmetic operations.

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

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

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

Q. Is “grade-school” matrix multiplication algorithm asymptotically optimal?

Block matrix multiplication

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

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

Block 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.

n-by-n matrices

$$C = A \times B$$

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

½n-by-½n matrices

8 matrix multiplications (of ½n-by-½n matrices)

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

4 matrix additions (of ½n-by-½n matrices)

Running time. Apply Case 3 of the master theorem.

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

Strassen's trick

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

scalars

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

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

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

7 scalar multiplications

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

$= A_{11} \times (B_{12} - B_{22}) + (A_{11} + A_{12}) \times B_{22}$

$= A_{11} \times B_{12} + A_{12} \times B_{22}$. ✔

Strassen's trick

Key idea. Can multiply two ~~2-by-2~~ *n*-by-*n* matrices via 7 ~~scalar~~ ½*n*-by-½*n* matrix multiplications (plus 11 additions and 7 subtractions).

½*n*-by-½*n matrices*

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

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

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

7 matrix multiplications (of ½n-by-½n matrices)

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

$= A_{11} \times (B_{12} - B_{22}) + (A_{11} + A_{12}) \times B_{22}$

$= A_{11} \times B_{12} + A_{12} \times B_{22}$. ✔

Strassen's algorithm

assume n is a power of 2

STRASSEN(n, A, B)

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

Partition A and B into ½n-by-½n 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$(n / 2, (A_{11} - A_{21}), (B_{11} + B_{12}))$.

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

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

$C_{12} = P_1 + P_2$.

$C_{21} = P_3 + P_4$.

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

$\Theta(n^2)$

RETURN C.

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

Analysis of Strassen's algorithm

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 $4.7 \cdot n^{\log 7}$ 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 $n^{\log 7}$ arithmetical operations.

Analysis of Strassen's algorithm

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 zeros to be n'-by-n', where $n \le n' < 2n$ and n' is a power of 2.

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

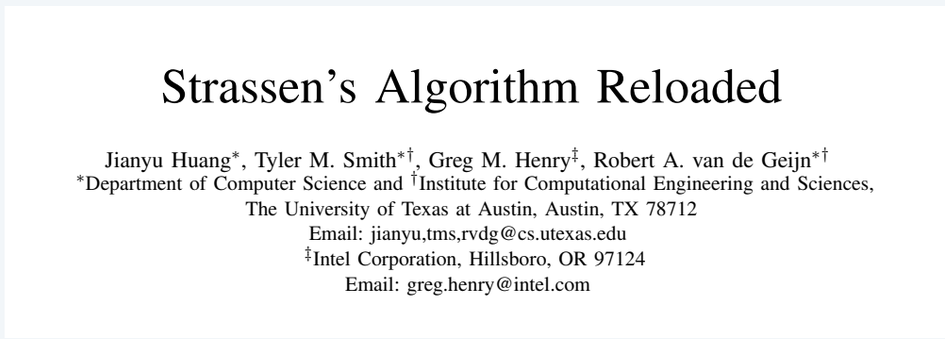
Strassen's algorithm: practice

Implementation issues.

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

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

- Apple reports 8x speedup when $n \approx 2{,}048$.
- Range of instances where it's useful is a subject of controversy.



Strassen's Algorithm Reloaded

Jianyu Huang*, Tyler M. Smith*†, Greg M. Henry‡, Robert A. van de Geijn*†
*Department of Computer Science and †Institute for Computational Engineering and Sciences,
The University of Texas at Austin, Austin, TX 78712
Email: jianyu,tms,rvdg@cs.utexas.edu
‡Intel Corporation, Hillsboro, OR 97124
Email: greg.henry@intel.com

Divide-and-conquer II: quiz 4

Suppose that you could multiply two 3-by-3 matrices with 21 scalar multiplications. How fast could you multiply two n-by-n matrices?

A. $\Theta(n^{\log_3 21})$

B. $\Theta(n^{\log_2 21})$

C. $\Theta(n^{\log_9 21})$

D. $\Theta(n^2)$

Divide-and-conquer II: quiz 5

Is it possible to multiply two 3-by-3 matrices using only 21 scalar multiplications?

A. Yes.

B. No.

C. Unknown.

Fast matrix multiplication: theory

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

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

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

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

Begun, the decimal wars have. [Pan 1978, Bini et al., Schönhage, ...]

- Two 70-by-70 matrices with 143,640 scalar multiplications. $O(n^{2.7962})$
- Two 48-by-48 matrices with 47,217 scalar multiplications. $O(n^{2.7801})$
- A year later. $O(n^{2.7799})$
- December 1979. $O(n^{2.521813})$
- January 1980. $O(n^{2.521801})$

History of arithmetic complexity of matrix multiplication

year	algorithm	arithmetic operations
1858	**“grade school”**	$O(n^3)$
1969	**Strassen**	$O(n^{2.808})$
1978	**Pan**	$O(n^{2.796})$
1979	**Bini**	$O(n^{2.780})$
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	**Strother**	$O(n^{2.3737})$
2011	**Williams**	$O(n^{2.372873})$
2014	**Le Gall**	$O(n^{2.372864})$
		$O(n^{2+\varepsilon})$

galactic algorithms

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

Numeric linear algebra reductions

Matrix multiplication. Given two n-by-n matrices, compute their product.

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	$\lvert A \rvert$	$\Theta(MM(n))$
rank	$rank(A)$	$\Theta(MM(n))$
system of linear equations	$Ax = b$	$\Theta(MM(n))$
LU decomposition	$A = L\,U$	$\Theta(MM(n))$
least squares	$\min \lVert Ax - b \rVert_2$	$\Theta(MM(n))$

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