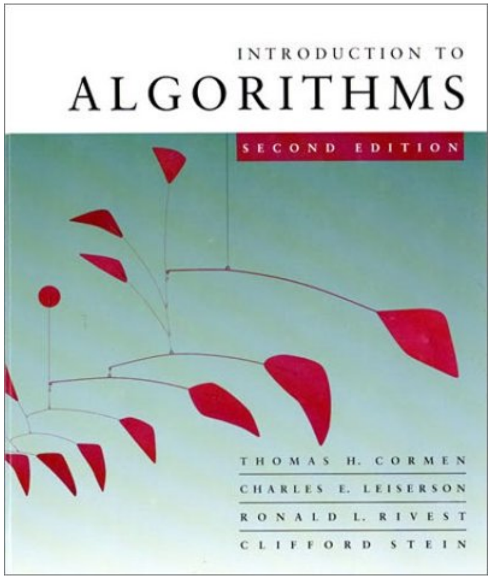
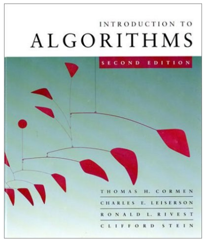
Introduction to Algorithms

6.006



Lecture 23

Jelani Nelson



Menu

3.141592653589793238462643383279502884197169399375105820974944592307816406286208998628034825342117067982148086513282306647093844609550582231725359408128481117450284102701938521105559644622948954930381964428810975665933446128475648233786783165271201909145648566923460348610454326648213393607260249141273724587006…

- "Numerics" - algorithms for operations on large numbers
  - Cryptography, simulations, etc
- Operations
  - Addition
  - Multiplication
  - Division
  - Matrix multiplication

Division a/b

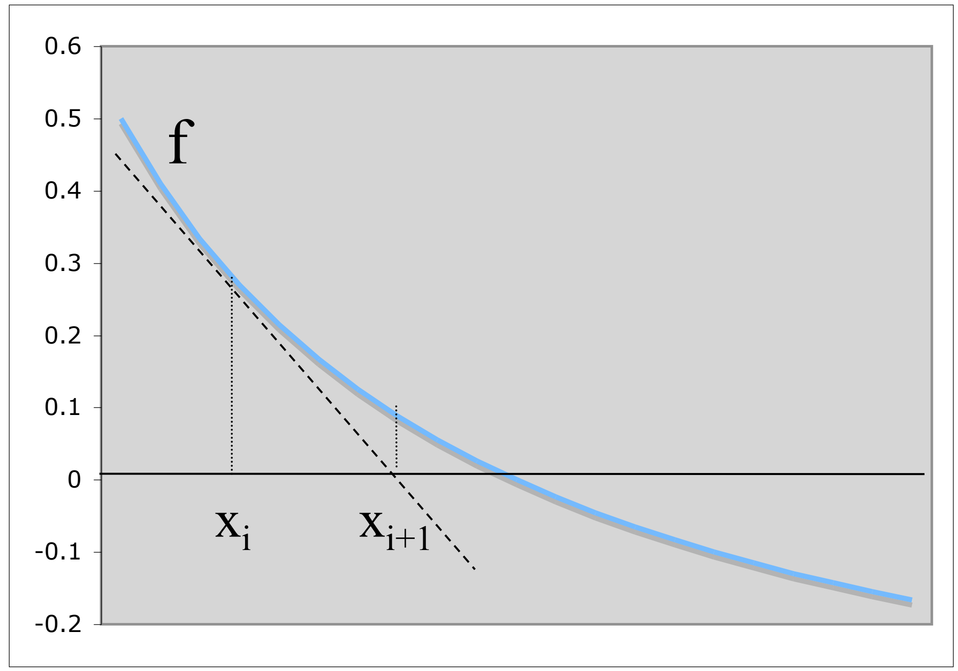
- Inversion:
  - Given positive n-digit number b (radix r)
  - Compute $1/b$

    I.e., for some $R=r^k$, compute $\lfloor R/b \rfloor$
- Iterative algorithm:
  - Start from some x_0
  - Keep computing x_{i+1} from x_i
  - Show this converges to the answer

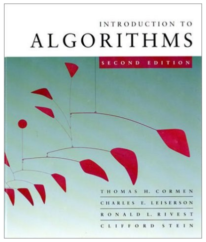
Newton's method

- Iterative approach to solving $f(x)=0$
  - In our case $f(x)=1/x-b/R$
- Iterative step:
  - Find a line $y= f(x_i)+f'(x_i)(x- x_i)$ tangent to $f(x)$ at x_i
  - Set x_{i+1} to the solution of

$$f(x_i)+f'(x_i)(x- x_i) =0$$



I.e., $x_{i+1}=x_i-f(x_i)/f'(x_i)$



Division: algorithm

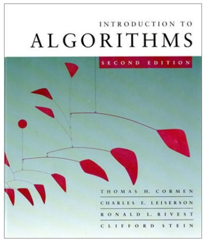
- Want to solve $f(x)=1/x-b/R=0$
- We have $f'(x)=-1/x^2$
- Iterative step $x_{i+1}=x_i-f(x_i)/f'(x_i)$ yields

$$x_{i+1}=x_i+ x_i^2\,(1/x_i - b/R)$$

I.e.,

$$x_{i+1}=2x_i - x_i^2 b/R$$

- Only O(1) multiplications, shifts and subtractions
- Convergence ?

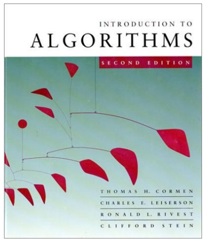


Convergence of $x_{i+1}=2x_i - x_i^2 b/R$

- Assume $x_i = R/b\,(1+e_i)$, e_i =error
- Assumptions:
  - $|e_i|$ is small
  - Ignore the round-off errors caused by the "/R" operation
- How does each iteration affect e_i ?

$$x_{i+1}=2x_i - x_i^2 b/R$$
$$= 2R/b\,(1+e_i) - [R/b\,(1+e_i)]^2\, b/R$$
$$= R/b[2+2e_i -1 -2e_i - e_i^2]$$
$$= R/b[1- e_i^2]$$
$$= R/b[1+ e_{i+1}], \text{ where } e_{i+1}=- e_i^2$$

- We have $|e_{i+1}|=|e_i|^2$ - "quadratic" convergence



Quadratic convergence $|e_{i+1}|=|e_i|^2$

- If we start close enough (say, $|e_0|<1/2$), then the convergence is very fast:

  $|e_1| \le |e_0|^2$

  $|e_2| \le |e_0|^{2*2}$

  …

  $|e_i| \le |e_0|^{2^\wedge i} < 1/2^{2^\wedge i}$

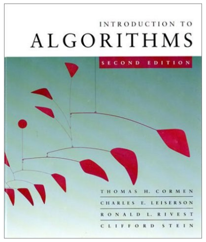
- To make sure we get k digits of precision, we need

  $|e_i| \quad < 1/r^k$

  $1/2^{2^\wedge i} < 1/r^k$

  $i \; > \log_2 (k \log_2 r)$

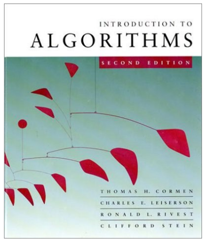
- What if $|e_0|>1/2$?
  - Heuristically, can apply the same algorithm
  - Analysis much more complicated
  - Does not always converge



General method

- E.g., square roots
  - $f(x)=x^2 - a$
  - $x_{i+1}= x_i -(x_i^2 -a) /2 x_i$

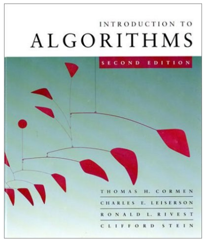
    $= (x_i +a/x_i)/2$
- Only O(1) multiplications, subtractions and divisions



Matrix multiplication

$$\left.\begin{array}{ll}\textbf{Input:} & A = [a_{ij}],\ B = [b_{ij}]. \\ \textbf{Output:} & C = [c_{ij}] = A \cdot B.\end{array}\right\}\ i, j = 1, 2, \ldots, n.$$

$$c_{ij} = \sum_k a_{ik} b_{kj}$$



Standard algorithm

for $i \leftarrow 1$ **to** n

  do for $j \leftarrow 1$ **to** n

    do $c_{ij} \leftarrow 0$

      for $k \leftarrow 1$ **to** n

        do $c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}$

Running time $= \Theta(n^3)$

Divide-and-conquer algorithm

IDEA:

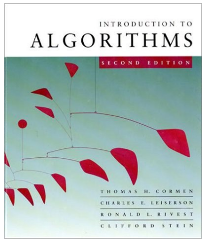
$n \times n$ matrix = 2×2 matrix of $(n/2) \times (n/2)$ submatrices:

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix}$$

$$C \quad = \quad A \quad \cdot \quad B$$

$$\left.\begin{aligned} r &= ae + bg \\ s &= af + bh \\ t &= ce + dg \\ u &= cf + dh \end{aligned}\right\}$$

8 mults of $(n/2) \times (n/2)$ submatrices
4 adds of $(n/2) \times (n/2)$ submatrices



Analysis of D&C algorithm

$$T(n) = 8\,T(n/2) + \Theta(n^2)$$

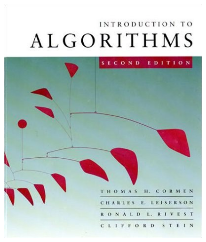
 # submatrices

submatrix size

work adding submatrices

$$n^{\log_b a} = n^{\log_2 8} = n^3 \;\Rightarrow\; \text{CASE 1} \;\Rightarrow\; T(n) = \Theta(n^3).$$

No better than the ordinary algorithm.



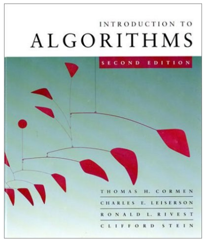
Strassen's idea (1969)

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix}$$

- Multiply 2×2 matrices with only 7 recursive mults.

$$\begin{aligned}
P_1 &= a \cdot (f - h) \\
P_2 &= (a + b) \cdot h \\
P_3 &= (c + d) \cdot e \\
P_4 &= d \cdot (g - e) \\
P_5 &= (a + d) \cdot (e + h) \\
P_6 &= (b - d) \cdot (g + h) \\
P_7 &= (a - c) \cdot (e + f)
\end{aligned}$$

$$\begin{aligned}
r &= P_5 + P_4 - P_2 + P_6 \\
s &= P_1 + P_2 \\
t &= P_3 + P_4 \\
u &= P_5 + P_1 - P_3 - P_7
\end{aligned}$$

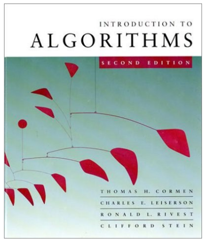


Strassen's idea

- Multiply 2×2 matrices with only 7 recursive mults.

$$
\begin{aligned}
P_1 &= a \cdot (f - h) \\
P_2 &= (a + b) \cdot h \\
P_3 &= (c + d) \cdot e \\
P_4 &= d \cdot (g - e) \\
P_5 &= (a + d) \cdot (e + h) \\
P_6 &= (b - d) \cdot (g + h) \\
P_7 &= (a - c) \cdot (e + f)
\end{aligned}
$$

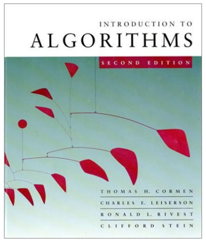
$$
\begin{aligned}
r &= P_5 + P_4 - P_2 + P_6 \\
&= (a + d)(e + h) \\
&\quad + d(g - e) - (a + b)h \\
&\quad + (b - d)(g + h) \\
&= ae + ah + de + dh \\
&\quad + dg - de - ah - bh \\
&\quad + bg + bh - dg - dh \\
&= ae + bg
\end{aligned}
$$



Strassen's algorithm

1. ***Divide:*** Partition A and B into $(n/2)\times(n/2)$ submatrices. Form terms to be multiplied using $+$ and $-$.
2. ***Conquer:*** Perform 7 multiplications of $(n/2)\times(n/2)$ submatrices recursively.
3. ***Combine:*** Form C using $+$ and $-$ on $(n/2)\times(n/2)$ submatrices.

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



Analysis of Strassen

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

$$n^{\log_b a} = n^{\log_2 7} \approx n^{2.81} \;\Rightarrow\; \text{CASE 1} \;\Rightarrow\; T(n) = \Theta(n^{\lg 7}).$$

Best to date (of theoretical interest only): $\Theta(n^{2.376})$.