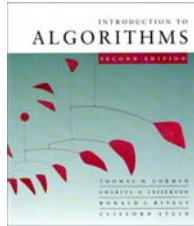




## CS 3343 -- Spring 2006



### Matrix Multiplication

Carola Wenk

Slides courtesy of Charles Leiserson with small changes by Carola Wenk

2/16/06

CS 3343 Analysis of Algorithms

1



## Powering a number

(a bit easier than the recursive mystery question on the homework)

**Problem:** Compute  $a^n$ , where  $n \in \mathbf{N}$ .

**Naive algorithm:**  $\Theta(n)$ .

**Divide-and-conquer algorithm:** (recursive squaring)

$$a^n = \begin{cases} a^{n/2} \cdot a^{n/2} & \text{if } n \text{ is even;} \\ a^{(n-1)/2} \cdot a^{(n-1)/2} \cdot a & \text{if } n \text{ is odd.} \end{cases}$$

$$T(n) = T(n/2) + \Theta(1) \Rightarrow T(n) = \Theta(\log n).$$

2/16/06

CS 3343 Analysis of Algorithms

2



## Matrix multiplication

**Input:**  $A = [a_{ij}], B = [b_{ij}]$ .  
**Output:**  $C = [c_{ij}] = A \cdot B$ .  
}  $i, j = 1, 2, \dots, n$ .

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

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

2/16/06

CS 3343 Analysis of Algorithms

3



## 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 =  $\Theta(n^3)$

2/16/06

CS 3343 Analysis of Algorithms

4



## Divide-and-conquer algorithm

### IDEA:

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

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

$$C = A \cdot B$$

$$\left. \begin{aligned} r &= a \cdot e + b \cdot g \\ s &= a \cdot f + b \cdot h \\ t &= c \cdot e + d \cdot h \\ u &= c \cdot f + d \cdot g \end{aligned} \right\}$$

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

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



## Analysis of D&C algorithm

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

# 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

- Multiply  $2 \times 2$  matrices with only 7 recursive mults.

$$P_1 = a \cdot (f - h)$$

$$P_2 = (a + b) \cdot h$$

$$P_3 = (c + d) \cdot e$$

$$P_4 = d \cdot (g - e)$$

$$P_5 = (a + d) \cdot (e + h)$$

$$P_6 = (b - d) \cdot (g + h)$$

$$P_7 = (a - c) \cdot (e + f)$$

$$r = P_5 + P_4 - P_2 + P_6$$

$$s = P_1 + P_2$$

$$t = P_3 + P_4$$

$$u = P_5 + P_1 - P_3 - P_7$$

7 mults, 18 adds/subs.

**Note:** No reliance on commutativity of mult!



## Strassen's idea

- Multiply  $2 \times 2$  matrices with only 7 recursive mults.

$$P_1 = a \cdot (f - h)$$

$$P_2 = (a + b) \cdot h$$

$$P_3 = (c + d) \cdot e$$

$$P_4 = d \cdot (g - e)$$

$$P_5 = (a + d) \cdot (e + h)$$

$$P_6 = (b - d) \cdot (g + h)$$

$$P_7 = (a - c) \cdot (e + f)$$

$$s = P_1 + P_2$$

$$= a \cdot (f - h) + (a + b) \cdot h$$

$$= af - ah + ah + bh$$

$$= af + bh$$



## Strassen's algorithm

- 1. Divide:** Partition  $A$  and  $B$  into  $(n/2) \times (n/2)$  submatrices. Form  $P$ -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

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

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

The number 2.81 may not seem much smaller than 3, but because the difference is in the exponent, the impact on running time is significant. In fact, Strassen's algorithm beats the ordinary algorithm on today's machines for  $n \geq 30$  or so.

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