

CSE 102 5-9-24

11

Strassen's Algorithm (4.2 p. 75-83)

Problem: multiply 2 $n \times n$ sq. matrices

$$\begin{array}{ccccc} C & = & A & \cdot & B \\ \uparrow & & \uparrow & & \uparrow \\ n \times n & & n \times n & & n \times n \end{array}$$

$$C = (c_{ij})_{1 \leq i \leq n, 1 \leq j \leq n}$$

where

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

Basic op. : multiplication of elements.

Routine : $\Theta(n^3)$

Recursive solution

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

\nearrow
 $\frac{n}{2} \times \frac{n}{2}$

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$

$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$

$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$

$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

combine 4 $\frac{n}{2} \times \frac{n}{2}$ submatrices into

$$C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

Exercise

- prove that this works!
- write pseudo-code.

note:

recursion to case $n=1$
 requires $\log_2 n$ is an
 exact power of 2.

let

$T(n) = \# \text{ real multiplications}$
on $n \times n$ arrays (n is
a power of 2.)

observe

$$T(n) = \begin{cases} 1 & n=1 \\ 8T(\frac{n}{2}) + 1 & \end{cases}$$

↑
const. cost of dividing &
combining. Book has n^2 ,
since they count additions.

By case 1 of master theorem

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

Volker Strassen: showed can

compute $C_{11}, C_{12}, C_{21}, C_{22}$

using only 7 $\frac{n}{2} \times \frac{n}{2}$ matrix

multiplications

Resulting in

$$n=1$$

$$T(n) = \begin{cases} 1 \\ 7T\left(\frac{n}{2}\right) + 1 \end{cases}$$

By master theorem

$$T(n) = \Theta(n^{\log_2 7})$$

$$= \Theta(n^{2.8073...})$$

$$= O(n^3)$$

□

What if n is not an exact power of 2?

Augment A and B with 0 rows and 0 columns to A' and B' of size N , which is a power of 2

$$A' = \begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix}, B' = \begin{pmatrix} B & 0 \\ 0 & 0 \end{pmatrix}$$

Exercise

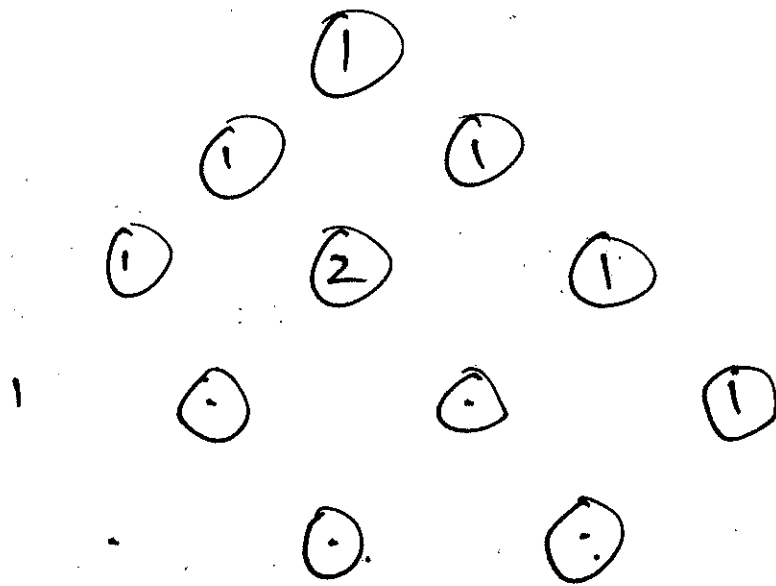
let N the smallest power
of 2 greater than or equal
to n . show that

$$N^{\lg(7)} = \Theta(n^{\lg(7)})$$

so augmentation has no (asymptotic)
cost.

Dynamic Programming: Handout.

Ex compute $\binom{n}{k}$



$$\binom{n}{k}$$



$$\text{row} = n$$

$$\text{col} = k$$