Polynomial time algorithm

$r = \frac{p}{q}$

size(r) = size(p) + size(q) where $\frac{p}{q} = r$ $q \in \mathbb{N}_+$, $\gcd(p,q) = 1$

Definition

An algorithm is polynomial time, if there exists a constant k such that the algorithm performs $O(n^k)$ operations on rational numbers whose size is bounded by $O(n^k)$. Here n is the number of bits that encode the input of the algorithm. We say that the algorithm runs in time $O(n^k)$.

1110101, 0110, 1101
INPUT

lenght of input = length of sequence of bits.

Example: Euclidean algorithm

$q \leq a$
$r \leq a/2$ } Numbers remain of size $\leq$ size(a)

$d \mid a,\ d \mid b$
$\Leftrightarrow d \mid b,\ d \mid r$

Input: Integers $a \geqslant b \geqslant 0$ not both equal to zero
Output: The greatest common divisor $\gcd(a, b)$

if $(b = 0)$ **return** a
else
Compute $q, r \in \mathbb{N}$ with $b > r \geqslant 0$ and $a = q \cdot b + r$ (division with remainder)
return $\gcd(b, r)$

Arithmetic Operation.

Second recursive call
$\gcd(r, r')$
now: $r \leq a/2$.
Every second iteration, the first argument is halved.
$b > r$

Analysis: Lemma: $r \leq a/2$

Proof: $a = q \cdot b + r$ $q \geqslant 1$ if $r > a/2$ then $a = q \cdot b + r \geqslant 1 \cdot b + r$
$> r + r \geq a$ ↯

Analysis

First argument ≥ 1. How to many recursive calls.

2. i recursive calls: First argument $\leq \frac{a}{2^i}$

$$\frac{a}{2^i} \geq 1 \iff a \geq 2^i$$

$$\iff \log_2 a \geq i$$

$\Rightarrow$ Number of recursive calls $\leq 2 \cdot \log_2(a) = O(\text{size}(a))$

$n = \text{size}(a) + \text{size}(b)$

Conclusion: Euclidean Algorithm runs in time $O(n)$

Determinant

Input: $A \in \mathbb{Q}^{n \times n}$
Output: $\det(A)$
if $(n = 1)$
  return a_{11}
else
  $d := 0$
  for $j = 1, \ldots, n$
    $d := (-1)^{1+j} \cdot \det(A_{1j}) + d$
  return d

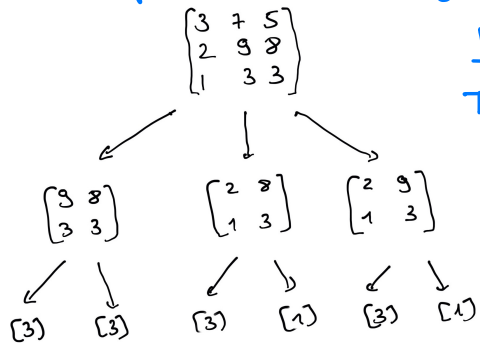
a_{1j} *

recursive call.

$$\det \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} = a_{11} \cdot \det(A_{11}) - a_{12} \det(A_{12}) + \cdots \pm (-1)^n \cdot \det(A_{1n})$$

Analysis

$T(n)$ is number of arithmetic operations that this algorithm performs.



$n \geq 2$

$T(n) \geq n \cdot T(n-1)$

$\geq n \cdot (n-1) \cdot T(n-2)$

$\vdots$

$\geq n! \geq 2^{n \log n}$

exponential time Alg.

Figure: An example of the recursion tree. The tree corresponds to the run of the algorithm on input $\begin{pmatrix} 3 & 7 & 5 \\ 2 & 9 & 8 \\ 1 & 3 & 3 \end{pmatrix}$.

not polynomial!

Gaussian elimination

Input: $A \in \mathbb{Q}^{m \times n}$

Output: A' in row echelon form such that there exists an invertible $Q \in \mathbb{Q}^{m \times m}$ such that $Q \cdot A = A'$.

$A' := A$

$i := 1$

while $(i \leq m)$

find minimal $1 \leq j \leq n$ such that there exists $k \geqslant i$ such that $a'_{kj} \neq 0$

If no such element exists, then **stop**

swap rows i and k in A'

for $k = i + 1, \ldots, m$

subtract (a'_{kj}/a'_{ij}) times row i from row k in A'

$i := i + 1$

Analysis:

$O(m^2 \cdot n)$ arithmetic operations

Question: How large can numbers become?

Without loss of Generality. Q. $A = \begin{pmatrix} \square & \square \\ 0 & D \end{pmatrix}$

Need to argue $\text{size}(e_{ij})$ is small.

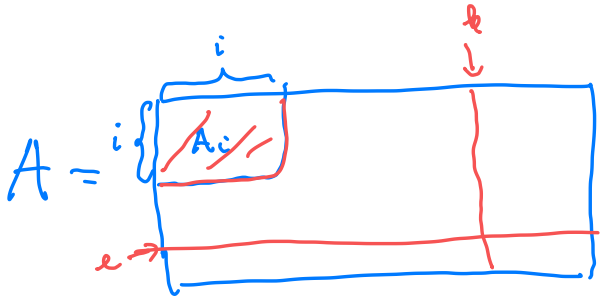
at step i:

$$\begin{bmatrix} a_{11}' & \cdots & \cdots \\ & \ddots & \vdots \\ 0 & & a_{ii}' \end{bmatrix} \quad \begin{matrix} D \\ \\ E \; e_{ek} \end{matrix}$$

OBSERVATION:

$$\det(A_i) = a_{11}' \cdot \ldots \cdot a_{ii}'$$

A_{ek} be the $(i+1)\times(i+c)$ matrix stemming from A by choosing rows $1,\ldots,i,k$ & cols $1,\ldots,i,i+c$

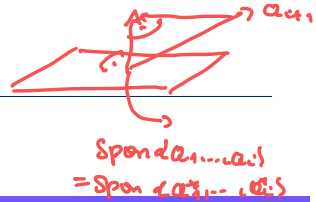


$\in \mathbb{Z}^{m\times n}$

$\det(A_{ek}) = \det(A_i) \cdot e_{ek}$

$\Leftrightarrow e_{ek} = \det(A_{ek}) / \det(A_i)$

Hadamard bound



$|\det(A)| = \prod \|a_i^*\|$

Theorem (Hadamard bound)

Let $A \in \mathbb{R}^{n \times n}$ be non-singular. Then

$$|\det(A)| \leq \prod_{i=1}^{n} \|a_i\|_2 \leq n^{n/2} \cdot B^n,$$

$\text{len} = \frac{\Delta_1}{\Delta_2}$

where B is upper bound on absolute values of entries of A.

Therfore: $\text{size}(\ell_{ek}) \leq \text{size}(\det(A_{ek})) + \text{size}(\det(A_{ii}))$

$$= O\left(\log_2 n^{n/2} \cdot B^n\right)$$

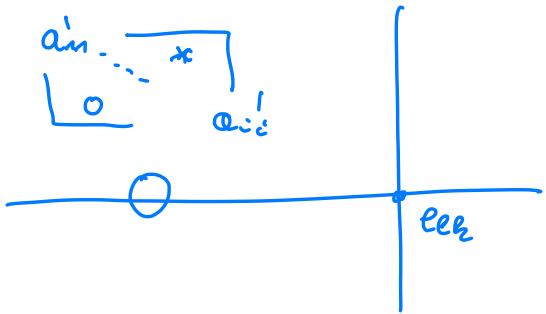
$$= O(n \cdot \log_2 B + n \cdot \log_2 n)$$

polynomial in total input length of A

$A = (101101, 0, \ldots, 1)$

n^2 strings of 0/1 each of most of length $\log_2(B)$

□


a_{in}
x
0
a_{ii}
e_{ek}


i


a_{in}
x
0
a_{ii}
0 ..... 0 e_{ek}

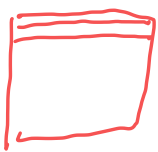
Analysis

Theorem

The Gaussian algorithm runs in polynomial time on input $A \in \mathbb{Z}^{m \times n}$. More precisely, the rational numbers produced in the algorithm can be maintained to be ratios of sub-determinants of A' and are thus of polynomial binary encoding length.

$A \in \mathbb{R}^{n \times n}$ $\quad b \in \mathbb{R}^n$

$A \cdot b$ $\quad \Theta(n^2)$ opérations



$\Theta(n)$

$A, B \in \mathbb{R}^{n \times n}$ $\quad A \cdot B$ en calculant

$A \cdot b_i$ $\quad i = 1, \dots, n$ où b_i i-ième colonne de B.

$\Rightarrow n \cdot \Theta(n^2) = \Theta(n^3)$

Matrix multiplication

Suppose $n = 2^k$

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

If we split the matrices A and B into 4 $n/2 \times n/2$ matrices

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \text{ and } B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \tag{6}$$

Then

$$A \cdot B = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} \cdot B_{11} + A_{12} \cdot B_{21} & A_{11} \cdot B_{12} + A_{12} \cdot B_{22} \\ A_{21} \cdot B_{11} + A_{22} \cdot B_{21} & A_{21} \cdot B_{12} + A_{22} \cdot B_{22} \end{pmatrix}.$$

Analysis: $T(n)$ # of arithmetic operations.

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

$T(n) \geq n^2 + 8 \cdot T(n/2) \geq n^2 + 8\left(\left(\frac{n}{2}\right)^2 + 8 \cdot T(n/4)\right)$

$T(n) \geq n^2 + 8 \cdot T(n/2) \geq n^2 + 8\left(\left(\frac{n}{2}\right)^2 + 8 \cdot T(n/4)\right)$

Strassen ↓ 7

$= n^2 + 2 \cdot n^2 + 8^2 \cdot T\left(\frac{n}{2^2}\right)$

$\geq n^2 + 2 \cdot n^2 + 8^2\left[\left[\frac{n}{2^2}\right]^2 + 8 \cdot T\left(\frac{n}{2^3}\right)\right]$

$= n^2 + 2 \cdot n^2 + \frac{8^2}{2^4} \cdot n^2 + 8^3 \cdot T\left(\frac{n}{2^3}\right)$

$= n^2 + 2n^2 + 2^2 n^2 + 8^3 \cdot T\left(\frac{n}{2^3}\right)$

$\geq \underbrace{n^2 + 2n^2 + 2^2 n^2 + \cdots + 2^i \cdot n^2}_{\times \log(n)} + 8^{i+1} T\left(\frac{n}{2^{i+1}}\right)$

$2^{i+1} = n$

$\log_2 n = i + 1$

$\geq n^3$ all for nothing!

Strassen: $T(n) = O(n^2) + 7 \cdot T(\frac{n}{2})$ is possible.

forget constant c.

$\leq c \cdot n^2 + 7\, T(n/2)$

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

$\leq n^2 + 7\left(\left(\frac{n}{2}\right)^2 + 7 \cdot T\left(\frac{n}{2^2}\right)\right)$

$\leq n^2 + 7 \cdot \frac{n^2}{2^2} + 7^2 \cdot \left(\frac{n}{2^2}\right)^2 + \dots$

$= n^2 \cdot \left(1 + \frac{7}{2^2} + \left(\frac{7}{2^2}\right)^2 + \dots \left(\frac{7}{2^2}\right)^{\log_2(n)}\right)$

$\sim \left(\frac{7}{4}\right)^{\log_2 n} \cdot n^2 = n^{\log_2 \frac{7}{4}} \cdot n^2$

$\log_2 \frac{7}{4} < 1$

$a^{\log_2 b} = b^{\log_2 a}$

Strassen's algorithm

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

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

$$\begin{aligned} M_1 &= (A_{11} + A_{22}) \cdot (B_{11} + B_{22}) \\ M_2 &= (A_{21} + A_{22}) \cdot B_{11} \\ M_3 &= A_{11} \cdot (B_{12} - B_{22}) \\ M_4 &= A_{22} \cdot (B_{21} - B_{11}) \\ M_5 &= (A_{11} + A_{12}) \cdot B_{22} \\ M_6 &= (A_{21} - A_{11}) \cdot (B_{11} + B_{12}) \\ M_7 &= (A_{12} - A_{22}) \cdot (B_{21} + B_{22}) \end{aligned}$$

$$\begin{aligned} C_{11} &= M_1 + M_4 - M_5 + M_7 \\ C_{12} &= M_3 + M_5 \\ C_{21} &= M_2 + M_4 \\ C_{22} &= M_1 - M_2 + M_3 + M_6. \end{aligned}$$

Exercise

$M_1 + M_4 - M_5 + M_2 = A_{11} \cdot B_{11} + A_{11} B_{22} + A_{21} B_{11} + A_{22} B_{22} + A_{12} B_{21} - A_{11} B_{11}$

$- A_{11} B_{22} - A_{22} B_{22} + A_{12} B_{21} + A_{12} B_{22} - A_{21} B_{11} - A_{22} B_{21}$

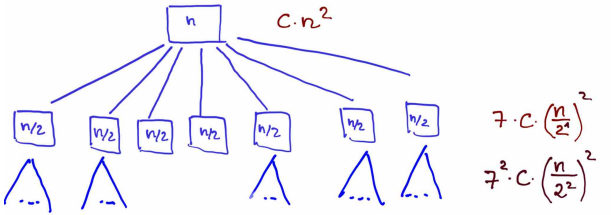
Strassen's algorithm

Input: Two $n \times n$ matrices A and B
Output: $C = \mathrm{FMM}(A, B)$, the product $A \cdot B$

if $n = 1$ return $a_{11} \cdot b_{11}$
else

$M_1 = \mathrm{FMM}(A_{11} + A_{22}, B_{11} + B_{22})$
$M_2 = \mathrm{FMM}(A_{21} + A_{22}, B_{11})$
$M_3 = \mathrm{FMM}(A_{11}, B_{12} - B_{22})$
$M_4 = \mathrm{FMM}(A_{22}, B_{21} - B_{22})$
$M_5 = \mathrm{FMM}(A_{11} + A_{12}, B_{22})$
$M_6 = \mathrm{FMM}(A_{21} - A_{11}, B_{11} + B_{12})$
$M_7 = \mathrm{FMM}(A_{12} - A_{22}, B_{21} + B_{22})$
Compute the matrices $C_{11}, C_{12}, C_{21}, C_{22}$ from $M_1, \ldots, M_7$
return C

Analysis



$$\text{Total: } c \sum_{i=0}^{\log_2 n} 7^i \cdot \frac{n^2}{4^i} = c \cdot n^2 \cdot \sum_{i=0}^{\log_2 n} \left(\frac{7}{4}\right)^i$$

$$\leq c \cdot n^2 \cdot \left(\frac{7}{4}\right)^{\log_2 n} = c \cdot n^{2 + \log_2 \left(\frac{7}{4}\right)} = c \cdot n^{2{,}807}$$

Running time

Theorem (Strassen)

Two $n \times n$ matrices can be multiplied in time (number of arithmetic operations) $O(n^{2+\log_2(7/4)})$.

One iteration of the simplex algorithm

Theorem

One iteration of the simplex algorithm requires a total number of $O(m \cdot n)$ operations on rational numbers whose size is polynomial in the input size.

$A \in \mathbb{R}^{m \times n}$, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$

$\max c^T x$
$Ax \leq b$

$B \subseteq \{1, \dots, m\}$ current basis. Suppose we have A_B^{-1} ← number of pg size

Compute: - $\lambda_B^T \cdot A_B = c^T \Leftrightarrow \lambda_B^T = c^T \cdot A_B^{-1}$ $O(n^2)$ operations.

- $A_B \cdot d = -e_i \Leftrightarrow d = A_B^{-1} \cdot (-e_i)$ $O(n)$ operations.

$i \in B$?

- $A \cdot d$ $O(m \cdot n)$ operations.

- exciting element j: $A \cdot x_B^* - b$ and ratios: $O(m \cdot n)$ operations.

given A_B^{-1} one iteration $O(m \cdot n^2)$ Arithmetic op.

i j

B

$B = \{ i, \dots \}$

SAME

$B_{NEW} = \{ j, \dots \}$

We have i and j as first basis element in old and new basis.

$$A_B^{-1} \cdot A_B = I_n$$

$$A_{B_{NEW}} A_B^{-1} = \begin{pmatrix} c_1 & \cdots & c_n \\ 0 & 1 & 0 \cdots 0 \\ 0 & 0 & 1 \, 0 \cdots \\ 0 & \cdots & 0 \, 1 \end{pmatrix} \quad \text{ID.}$$

$A_{B_{NEW}}^{-1}$ can be obtained in $O(n^2)$ operations

Again in great detail next week!