# The Speed Limit of Multiplication.

**Ola Mæhlen**
**Institutt for matematiske fag**
**NTNU**

**"Forum for matematiske perler"**
**IMF, NTNU**
**06. Sept. 2019**

**In this presentation, numbers are represented in base 10.**

**Thus, computing a number means to calculate its digits in base 10.**

## Definitions

A multiplication algorithm $\mathscr{A}$ takes in two integers and computes their product. (Here in base 10.)

We let $M_{\mathscr{A}}(n)$ denote the maximum number of elementary steps (addition/multiplication of single digits, etc) needed for $\mathscr{A}$ to compute the product of two n-digit integers.

The complexity of $\mathscr{A}$ is the asymptotic behaviour of $M_{\mathscr{A}}(n)$ as n goes to infinity.

The complexity of multiplication is the optimal complexity of all possible multiplication algorithms.

We formally write $M(n)$ for the number of steps required by an "optimal multiplication algorithm" (i.e. of optimal complexity).

The complexity of multiplication is the asymptotic behaviour of $M(n)$.

## Why do we care about the complexity of multiplication?

**Multiplication is a fundamental building block in computation:**

| Operation | Algorithm | Complexity |
|---|---|---|
| Squaring | Optimal multiplication algorithm | $O(M(n))$ |
| Division | Newton-Raphson division | $O(M(n))$ |
| Square root (first n digits) | Newton's method | $O(M(n))$ |
| Greatest common divisor | Schönhage controlled Euclidean descent algorithm | $O(M(n)\log(n))$ |
| $\pi$ ( n decimal places) | Gauss-Legendre algorithm | $O(M(n)\log(n))$ |

**Note that the complexity of squaring cannot be much lower than half that of multiplication, as**

$$xy = \frac{(x+y)^2 - (x-y)^2}{4}$$

**and so one multiplication can be exchange for the cost of two squares (+ negligible extra steps).**

## Grade-school multiplication (GS)

**-The algorithm taught in school.**
**-Similar method used in ancient Egypt at least 4000 years ago.**
**-Has quadratic complexity i.e. $M_{GS}(n)$ behaves no better than $O(n^2)$ asymptotically.**

**Example with $n = 6$ and numbers $a = 249416$, $b = 133758$,**

$$
\begin{array}{rl}
 & 249416 \\
\times & 133758 \\
\hline
1995328 & \leftarrow 8\times 249416 \\
12470800 & \leftarrow 5\times 249416 \\
174591200 & \leftarrow 7\times 249416 \\
748248000 & \leftarrow 3\times 249416 \\
7482480000 & \leftarrow 3\times 249416 \\
24941600000 & \leftarrow 1\times 249416 \\
\hline
= \ 33361385328 &
\end{array}
$$

On $k$'th row: Multiply 1 digit with $n$-digit number, and add $k-1$ extra zeros to the result (shifts) requires $\approx n+k-1$ steps.
Lastly, we sum over $n$ numbers of length $\in [n,2n]$, requires $\approx n^2$ steps.
Number of elementary steps is approximately

$$M_{GS}(n) \approx \left(\sum_{k=1}^{n} n + (k-1)\right) + n^2 = \frac{5n^2}{2} - \frac{n}{2}$$

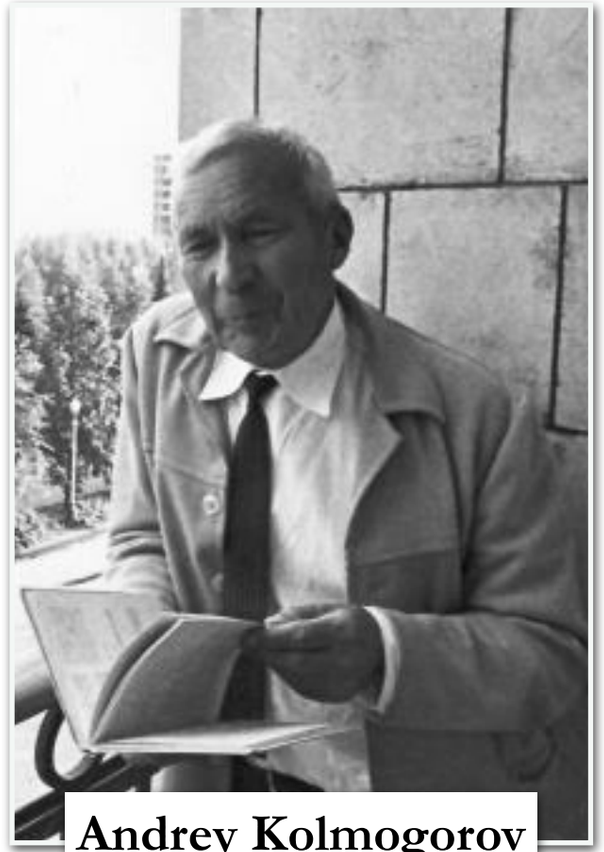
$$\Rightarrow M_{GS}(n) \approx n^2 .$$

**Consequently, the complexity of multiplication satisfies (at least)**

$$M(n) = O(n^2) .$$

## Kolmogorov's $n^2$-conjecture

**In 1956 Andrey Kolmogorov conjectured that the complexity of multiplication is quadratic, $M(n) \approx n^2$.**
**Later in 1960 he organised a seminar on problems in cybernetics at Moscow University; here he stated his conjecture.**


Andrey Kolmogorov

**Attending was the 23-year old student Anatoly Karatsuba.**
**A week's search later, he discovered the Karatsuba algorithm:**

$$M_{KA}(n) = O\left(n^{\log_2(3)}\right), \qquad \log_2(3) = 1.58496...$$

**disproving Kolmogorov's conjecture!**

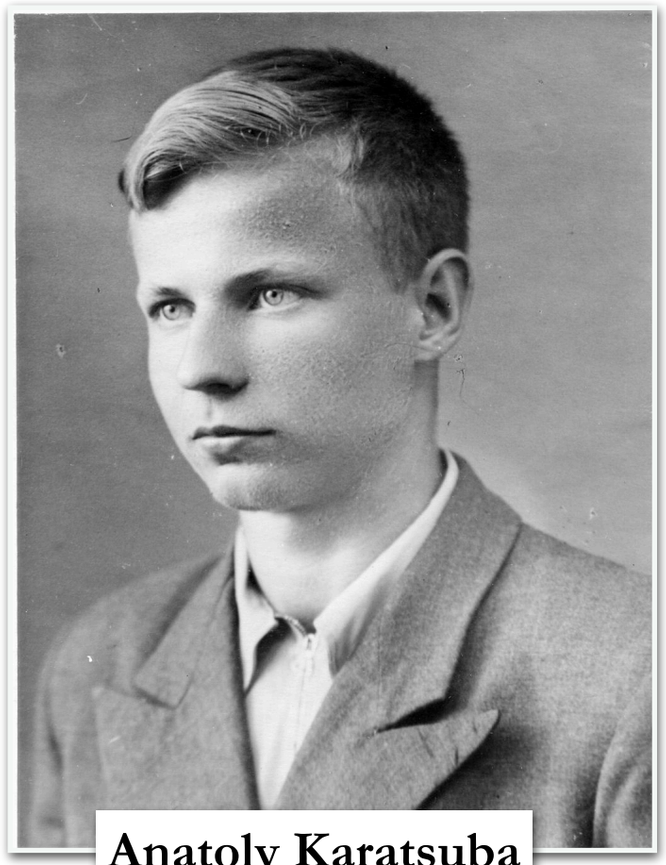
**After learning of this new method, Kolmogorov presented it at the next meeting… and then terminated the seminar.**

**In 1962, Kolmogorov wrote and published the article:**

A. Karatsuba and Yu. Ofman,
"*Multiplication of Multiplace Numbers on Automata*"

**Karatsuba first learned of the article when given its reprint.**

**The article spawned a new area of computation theory:**
**Fast multiplication algorithms.**


Anatoly Karatsuba

## Karatsuba algorithm (KA), 1960

**We consider the integers from earlier** $a = \overbrace{249}^{a_1}\overbrace{416}^{a_2}, \quad b = \overbrace{133}^{b_1}\overbrace{758}^{b_2}.$

**Thus:**

$$a = 249 \times 1000 + 416 = a_1 10^3 + a_2,$$
$$b = 133 \times 1000 + 758 = b_1 10^3 + b_2,$$
$$ab = a_1 b_1 10^6 + (a_1 b_2 + a_2 b_1)10^3 + a_2 b_2,$$

**Knowing** $a_1b_1$, $a_1b_2$, $a_2b_1$ **and** $a_2b_2$, **we could build** $ab$ **using only addition and shifts,**

$$M(n) < 4M(n/2) + O(n),$$
$$\Rightarrow M(n) = O(n^2)\,.$$

**Karatsuba realized that one can obtain the sum** $a_1b_2 + a_2b_1$, **without calculating the products** $a_1b_2$ **and** $a_2b_1$ **individually. Indeed, knowing** $a_1b_1$ **and** $a_2b_2$ **we need only one extra multiplication to attain this sum:**

$$a_1 b_2 + a_2 b_1 = (a_1 + a_2)(b_1 + b_2) - a_1 b_1 - a_2 b_2,$$

$$M(n) < 3M(n/2) + O(n), \quad \Rightarrow M(n) = O\left(n^{\log_2(3)}\right).$$

$\mathscr{A}_{KA}(a,b)$ : **if** $n = 1$, **return** $ab$.

**else set** $x = \mathscr{A}_{KA}(a_1, b_1)$,
$y = \mathscr{A}_{KA}(a_2, b_2)$,
$z = \mathscr{A}_{KA}(a_1 + a_2, b_1 + b_2)$,

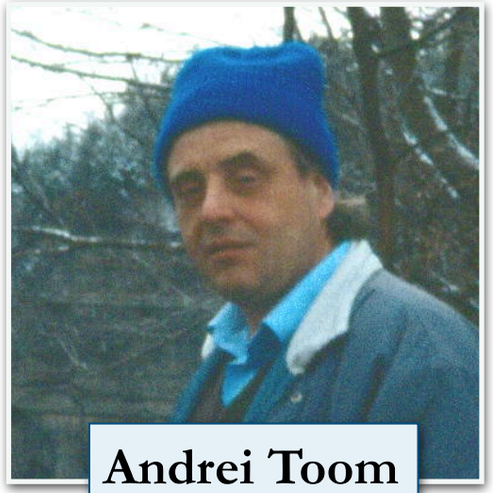
**and return** $x10^{2n} + (z - x - y)10^n + y$.

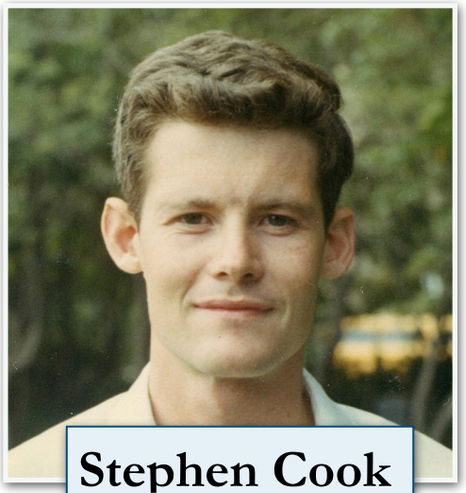
$$M_{KA}(n) = O\left(n^{\log_2(3)}\right).$$

**More efficient than grade-school multiplication around** $n > 60$.

## Toom-Cook algorithm (Tk), 1963

**Introduced by Andrei Toom, and further simplified by Stephen Cook .**
**A family of algorithms: Toom-$k$ (KA=Toom-2).**


Andrei Toom


Stephen Cook

**Example of Toom-3, with $n = 6$ :**

$$a = 249416 \;\rightarrow\; A(x) = 24x^2 + 94x + 16,$$
$$b = 133758 \;\rightarrow\; B(x) = 13x^2 + 37x + 58.$$

**Notice:**
$$C(100) = A(100)B(100) = ab\,.$$

**Want to calculate $C(x) = A(x)B(x)$. Being a polynomial of degree 4, we need 5 evaluations. Choose small values: $x = 2,1,0,-1,-2$.**

**Coefficients of $C(x)$**

**Each is a product of integers of $\approx n/3$ digits. Toom-3 is reapplied to evaluate these 5 products.**

$$\begin{bmatrix} C(2) \\ C(1) \\ C(0) \\ C(-1) \\ C(-2) \end{bmatrix} = \begin{bmatrix} 2^4 & 2^3 & 2^2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & -1 & 1 & -1 & 1 \\ (-2)^4 & (-2)^3 & (-2)^2 & -2 & 1 \end{bmatrix} \begin{bmatrix} c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix}$$

**Compact notation:**
$$\vec{C} = \Pi\, \vec{c}$$

**Calculating $\Pi^{-1}$(in advance), $\vec{c}$ is retrieved by a weighted sum of the elements of $\vec{C}$.**

$$M_{T3}(n) < 5M_{T3}(n/3) + O(n),$$

$$\Rightarrow M_{T3}(n) = O\left(n^{\log_3(5)}\right).$$

**Toom-$k$:** $M_{Tk}(n) < (2k-1)M_{Tk}(n/k) + O(n),$

$$\Rightarrow M_{Tk}(n) = O\left(n^{\log_k(2k-1)}\right).$$

**Toom-Cook can get "linear + $\varepsilon$" complexity, as** $\log_k(2k-1) \searrow 1, \quad k \to \infty$ .

## Some more definitions

Sequences are by default of length $N$.

Notation: $(x_k) = (x_k)_{k=0}^{N-1} = (x_0, x_1, \ldots, x_{N-1})$.

**Pointwise product:**

$$(x_k) \cdot (y_k) = (x_k y_k),$$

**Cost** $= O(NM(n))$

**Cyclic convolution:**

$$(x_k) * (y_k) = (\sum_j x_j y_{k-j}),$$

## Some more definitions

Sequences are by default of length $N$.
Notation: $(x_k) = (x_k)_{k=0}^{N-1} = (x_0, x_1, \ldots, x_{N-1})$.

Pointwise product:

$$(x_k) \cdot (y_k) = (x_k y_k),$$
$$\text{Cost} = O(NM(n))$$

Cyclic convolution:

$$(x_k) * (y_k) = (\sum_j x_j y_{k-j}),$$

**The index $k - j$ is considered modulo $N$.**

## Some more definitions

Sequences are by default of length $N$.
Notation: $(x_k) = (x_k)_{k=0}^{N-1} = (x_0, x_1, \ldots, x_{N-1})$.

### Discrete Fourier transform:

$$(\hat{x}_k) = \mathscr{F}\{(x_k)\} \qquad (x_k) = \mathscr{F}^{-1}\{(\hat{x}_k)\}$$

$$\Rightarrow \hat{x}_k = \sum_j x_j e^{-\frac{2\pi i}{N}kj}, \qquad \text{Cost} = O(N^2 M(n))$$

$$\Rightarrow x_k = \frac{1}{N}\sum_j \hat{x}_j e^{\frac{2\pi i}{N}kj}, \qquad \text{Cost} = O(N^2 M(n))$$

### Pointwise product:

$$(x_k)\cdot(y_k) = (x_k y_k), \qquad \text{Cost} = O(NM(n))$$

### Cyclic convolution:

$$(x_k) * (y_k) = (\sum_j x_j y_{k-j}), \qquad \text{Cost} = O(N^2 M(n))$$

### Convolution theorem:

$$(x_k) * (y_k) = \mathscr{F}^{-1}\{(\hat{x}_k)\cdot(\hat{y}_k)\}.$$

The expression $\mathbb{Z}[x]/\langle x^N - 1\rangle$ denotes the ring of polynomials with integer coefficients modulo $x^N - 1$, in other words, we have $x^N = 1$.

Elements of $\mathbb{Z}[x]/\langle x^N - 1\rangle$ are represented by polynomials of degree less than $N$.

| | | | |
|---|---|---|---|
| **Polynomials:** | $P(x)$ | $Q(x)$ | $P(x)Q(x)$ |
| **Coefficients:** | $(p_k)$ | $(q_k)$ | $(p_k) * (q_k)$ |

For polynomials $A(x), B(x)$ of degree less than $N/2$, the classical product coincides with the ring-product. The coefficients $(c_k)$ of $C(x) = A(x)B(x)$ can be calculated from:

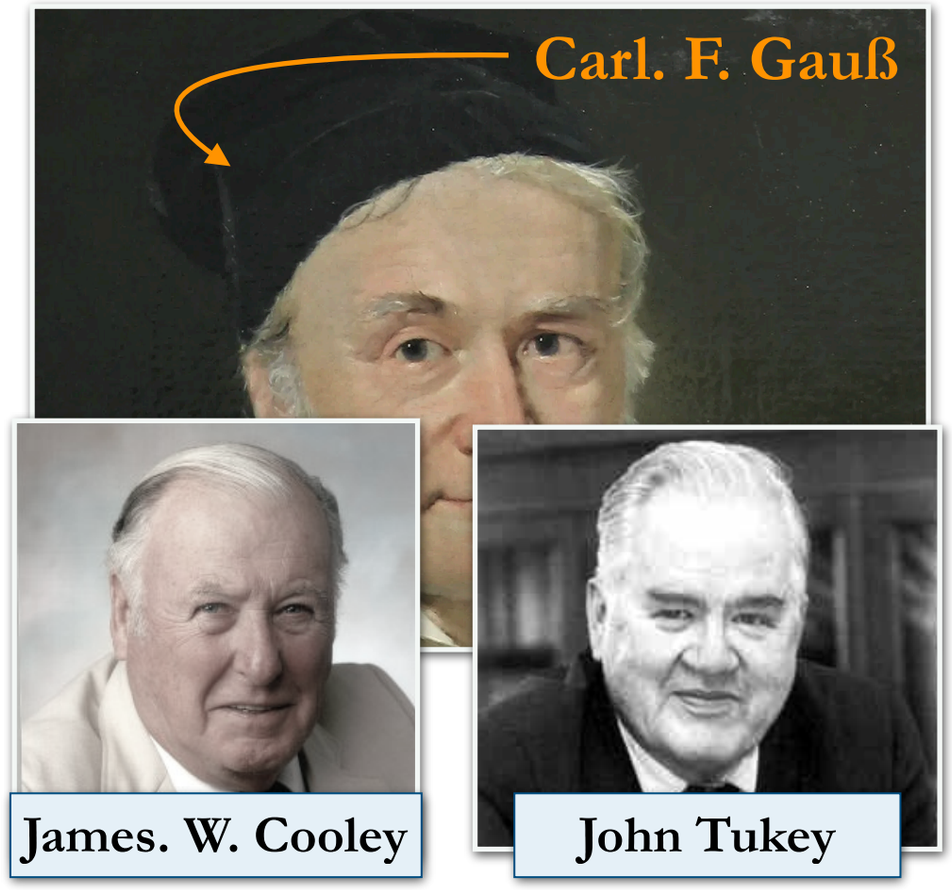
$$(c_k) = (a_k) * (b_k) = \mathscr{F}^{-1}\{(\hat{a}_k)\cdot(\hat{b}_k)\},$$

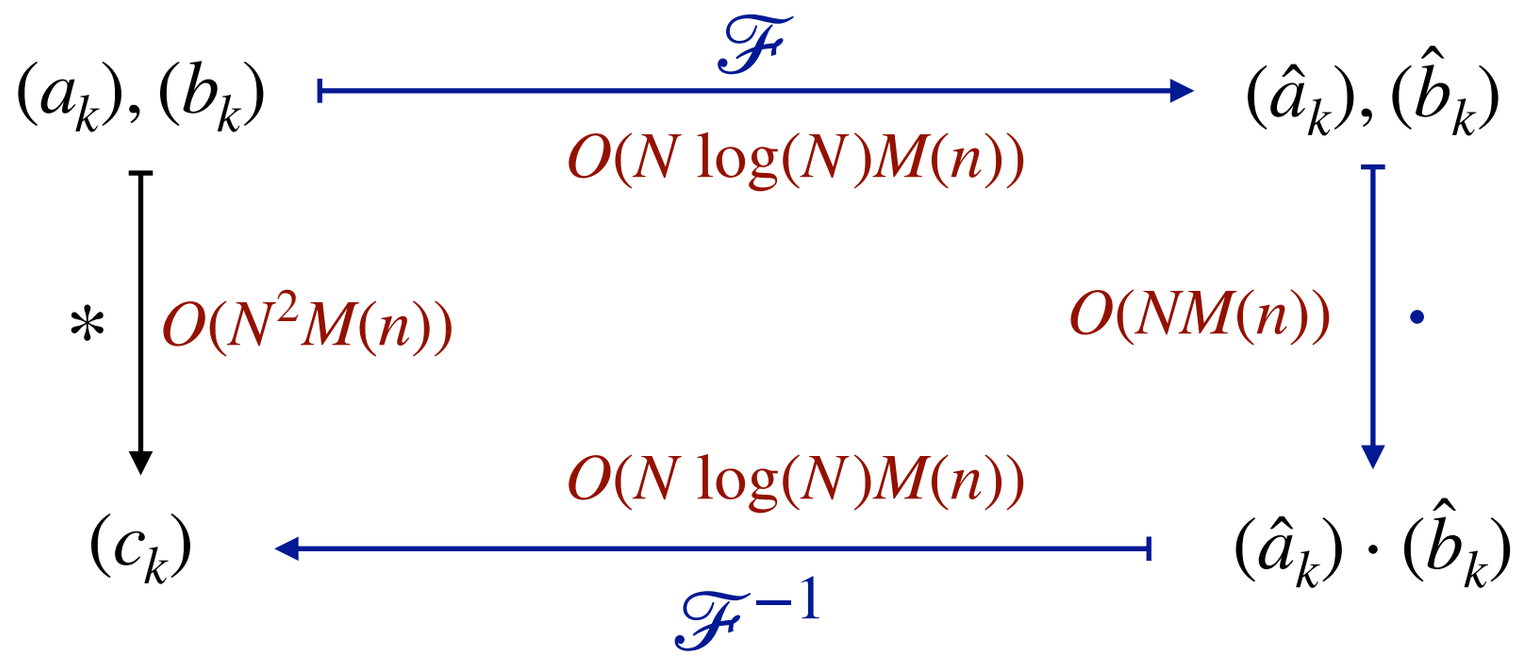
though seemingly not more efficient…

## Fast Fourier transform (FFT)

**A Fast Fourier transform is any discrete Fourier transform algorithm of complexity $O(N \log(N) M(n))$ and not $O(N^2 M(n))$.**

**Popularised in 1965, when the Cooley-Tukey FFT was introduced.**
**The same algorithm was known to Gauss, who used it to interpolate the trajectories of the asteroids Pallas and Juno. Unfortunately, his work was first published after his death and only in Latin.**



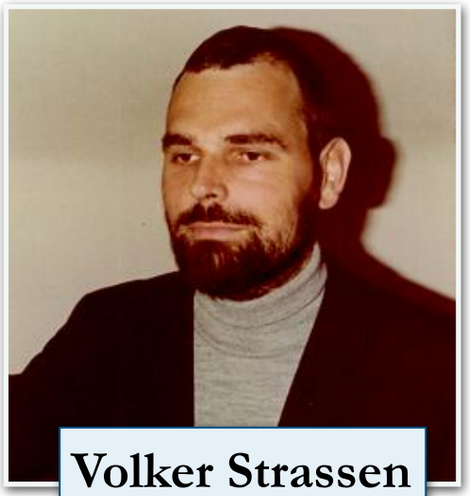




**Blue path calculates $(c_k)$ in $O(N \log(N) M(n))$ steps!**

## Schönhage-Strassen algorithm (SSA1, SSA2), 1971

**Two algorithms developed by Arnold Schönhage and Volker Strassen in 1971.**


Arnold Schönhage


Volker Strassen

**Description of SSA1 with numbers $a, b$ of $n$ digits:** (Set $N = n/\log(n)$)

$$a \to A(x) = a_{N-1}x^{N-1} + \ldots + a_0, \qquad a = A\left(10^{n/N}\right), \qquad (a_k) = (a_0, \ldots, a_{N-1}, 0, \ldots, 0).$$

$$b \to B(x) = b_{N-1}x^{N-1} + \ldots + b_0, \qquad b = B\left(10^{n/N}\right), \qquad (b_k) = (b_0, \ldots, b_{N-1}, \underbrace{0, \ldots, 0}).$$

**Consider $A(x), B(x)$ elements of $\mathbb{Z}[x]/\langle x^{2N} - 1\rangle$. Thus $(a_k), (b_k)$ ends with $N$ zeroes to represent the absence higher order terms. To attain $C(x) = A(x)B(x)$, we compute the coefficients with a FFT:**

$$\mathcal{F}^{-1}\{(\hat{a}_k) \cdot (\hat{b}_k)\} = (c_k)$$

**LHS contains $O(N\log(N))$ multiplications of $\approx n/N$-digit numbers. SSA1 is reapplied.**

**The final product is constructed from shifts and addition:**

$$ab = C\left(10^{n/N}\right) = \sum_{k=0}^{2N-1} c_k 10^{kn/N}.$$

$$M_1(n) = O\left(N\log(N)M_1(n/N)\right) = O\left(nM_1(\log(n))\right), \quad \Rightarrow M_1(n) = O\left(n\log(n)^{1+\varepsilon}\right)$$

**SSA2 is more famous, elegant and popular. Similar to SSA1, but uses the more 'natural' number theoretic transform instead of FFT. This algorithm attains better complexity:**

$$M_2(n) < 2\sqrt{n}\, M_2\left(\sqrt{n}\right) + O\left(n\log(n)\right), \quad \Rightarrow M_2(n) = O\left(n\log(n)\log\log(n)\right)$$

## The first Schönhage-Strassen multiplication algorithm in action:

$$249416 \times 133758 = R(100)$$

$$\underbrace{(24x^2 + 94x + 16)}_{P(x)} \times \underbrace{(13x^2 + 37x + 58)}_{Q(x)} = R(x)$$

$$\mathcal{F} \qquad\qquad \mathcal{F}^{-1}$$

$$\widehat{P} \cdot \widehat{Q} = \widehat{R}$$

Recursion happens here

**Schönhage-Strassen $n\log(n)$-conjecture, 1971**

**The complexity of SSA contains a large constant factor (efficient for $n > 10\,000$), but has the asymptotic behaviour:**

$$M(n) = O\left(n\log(n)\log\log(n)\right).$$

**Is this optimal?** **In their article, Schönhage and Strassen write:**

> **"We do not believe that the size of $n\log(n)\log\log(n)$ is optimal, but suspect this for the order of magnitude $n\log(n)$. "**

**Despite rapid progression from 1962-1971, not much improvement on fast multiplication happened for the next 36 years.**

**In 2007, a Swiss mathematician, Martin Fürer, discovered an algorithm whose complexity replaced $\log\log(n)$ with $2^{\log^*(n)}$. Unfortunately the complexity hides an enormous constant factor; the algorithm is estimated to be faster than SSA for 'astronomically' large data ($n > 10^{10^{4796}}$), making it a so called 'galactic' algorithm.**

**Still, the $n\log(n)$ seemed within reach…**

## Harvey-van der Hoeven algorithm, 2019

**By David Harvey and Joris van der Hoeven.**
**Preprint added to Hyper Articles en Ligne (HAL)**
**March 2019.**

**Of complexity $n\log(n)$.**

David Harvey

Joris van der Hoeven

**Key feature: arranging data in multiple dimensions.**

**When $N = p_1 p_2 \cdots p_d$ (distinct primes), the Chinese remainder theorem implies the ring isomorphism:**

$$\mathbb{Z}[x]/\langle x^N - 1\rangle \approx \mathbb{Z}[x_1, \ldots, x_d]/\langle x_1^{p_1} - 1\rangle \cdots \langle x_d^{p_d} - 1\rangle \quad \text{f.ex.} \quad x \leftrightarrow x_1 x_2 \cdots x_d$$

**Naturally re-arranges the coefficients from a sequence to a $d$-dimensional array:**

$$\begin{aligned} (c_k) &\leftrightarrow (c_{k_1,k_2,\ldots,k_d}), & 0 \le k_1 \le p_1 - 1,\\ &&\vdots\\ (a_k) * (b_k) &\leftrightarrow (a_{k_1,k_2,\ldots,k_d}) * (b_{k_1,k_2,\ldots,k_d}), & 0 \le k_d \le p_d - 1. \end{aligned}$$

**Seemingly no improvement: the convolutions are equally expensive, even when exploiting classical FFTs…**

**Harvey and van der Hoeven introduces two new devices.**

## Harvey-van der Hoeven algorithm, 2019

By David Harvey and Joris van der Hoeven.
Preprint added to Hyper Articles en Ligne (HAL)
March 2019.

Of complexity $n\log(n)$.


David Harvey


Joris van der Hoeven

Key feature: arranging data in multiple dimensions.

---

**Device 1:** **Efficient FFT for power-of-two sized arrays.**

$\bigotimes_{j=1}^{d} \mathbb{C}^{t_j}$ denotes the set of complexed valued $d$-dimensional arrays of sizes $t_1 \times t_2 \times \cdots \times t_d$ .

For powers of two $t_1, t_2, \ldots, t_d$, they construct a FFT

$$\bigotimes_{j=1}^{d} \mathbb{C}^{t_j} \xrightarrow{\mathscr{F}} \bigotimes_{j=1}^{d} \mathbb{C}^{t_j}$$

consisting of few medium-sized multiplications, instead of many small.

---

On the previous slide we established an isomorphism

$$\mathbb{C}^{N} \longleftrightarrow \bigotimes_{j=1}^{d} \mathbb{C}^{p_j}$$

but $p_1, p_2, \ldots, p_d$ are distinct primes $\Longrightarrow$ not powers of two. So how is this useful?

## Harvey-van der Hoeven algorithm, 2019

By David Harvey and Joris van der Hoeven.
Preprint added to Hyper Articles en Ligne (HAL)
March 2019.

**Of complexity $n\log(n)$.**


David Harvey


Joris van der Hoeven

**Key feature: arranging data in multiple dimensions.**

---

**Device 2:**

**They construct two cost-efficient maps, $\mathscr{A}, \mathscr{B}$,**

**Satisfying:** $\mathscr{F} = \mathscr{B} \circ \mathscr{F} \circ \mathscr{A}$

**Gaussian resampling.**

$$\bigotimes_{j=1}^{d} \mathbb{C}^{p_j} \underset{\mathscr{B}}{\overset{\mathscr{A}}{\rightleftarrows}} \bigotimes_{j=1}^{d} \mathbb{C}^{t_j}$$

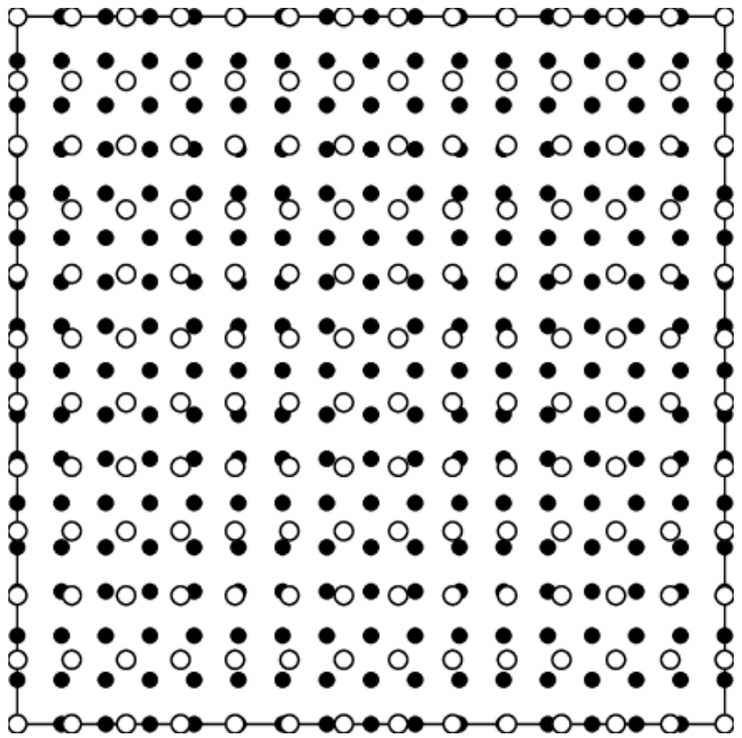
---

**Roughly how to construct $\mathscr{A}(u)$, from $u \in \bigotimes_{j=1}^{d} \mathbb{C}^{p_j}$ :**

**Consider $u$ as a $p_1 \times \cdots \times p_d$ grid, scaled to fit inside the $d$-dimensional unit torus $(\mathbb{R}/\mathbb{Z})^d$. A complex number is associated to each grid-point. We place a $d$-dimensional Gaussian at each grid-point, scaled with the associated number. We then superimpose a $t_1 \times t_2 \cdots \times t_d$ grid $v$ over $u$, and associate to each point of $v$ the "sum of the Gaussians" of $u$ evaluated at said point.**

$$v = \mathscr{A}(u)$$

**(Gaussians are convenient for two reasons: rapid decay and invariance under under $\mathscr{F}$.)**



$u = 11 \times 13$ grid (white)
$v = 16 \times 16$ grid (black)

## Harvey-van der Hoeven algorithm, 2019

**By David Harvey and Joris van der Hoeven.**
**Preprint added to Hyper Articles en Ligne (HAL)**
**March 2019.**

**Of complexity $n\log(n)$.**

David Harvey

Joris van der Hoeven

**Key feature: arranging data in multiple dimensions.**

$$P_d = \bigotimes_{j=1}^{d} \mathbb{C}^{p_j}, \qquad T_d = \bigotimes_{j=1}^{d} \mathbb{C}^{t_j}.$$

**The algorithm takes the red path in the commutative diagram:**

$$\begin{array}{ccccccccc}
(a_k),(b_k) \in \mathbb{C}^N \times \mathbb{C}^N & \xrightarrow{\sim} & P_d \times P_d & \xrightarrow{\mathscr{A}} & T_d \times T_d & \xrightarrow{\mathscr{F}} & T_d \times T_d \\
\downarrow * \quad \downarrow * & & \downarrow * & \text{Device 2} & \downarrow * & \text{Device 1} & \downarrow \cdot \\
(c_k) \in \mathbb{C}^N & \xleftarrow{\sim} & P_d & \xleftarrow{\mathscr{B}} & T_d & \xleftarrow{\mathscr{F}^{-1}} & T_d
\end{array}$$

$$M_{HH}(n) < Kn^{\left(\frac{d-1}{d}\right)} M_{HH}\left(n^{\frac{1}{d}}\right) + O\left(n\log(n)\right)$$

**(Where $K$ is independent of $d$.)**

**Choosing a $d > K$, we obtain:**

$$M_{HH}(n) = O\left(n\log(n)\right)$$

## True complexity of multiplication

We now know

$$M(n) = O\big(n\log(n)\big),$$

but is this optimal?

Notoriously difficult to prove lower bounds in computation theory.
(Ex. $P = NP$)

The conventional belief is that $n\log(n)$ is optimal, but experts have been wrong before.

# References

-A. A. Karatsuba (1995). "The Complexity of Computations". *Proceedings of the Steklov Institute of Mathematics*. **211**: 169–183.

-A. Schönhage and V. Strassen, "Schnelle Multiplikation großer Zahlen", *Computing* **7** (1971)

-David Harvey, Joris van der Hoeven. "Integer multiplication in time O(n log n)." 2019. ⟨hal-02070778⟩

-D. Harvey, J. van der Hoeven, and G. Lecerf, "Even faster integer multiplication", J. Complexity 36 (2016), 1–30. MR 3530637.

-P. Afshani, C. B. Freksen, L. Kamma, and K. G. Larsen, "Lower bounds for multiplication via network coding", 2019.

-Alexander Kruppa. "Speeding up integer multiplication and factorization". General Mathematics [math.GM]. Université Henri Poincaré - Nancy 1, 2010. English. ffNNT : 2010NAN10054ff. fftel01748662

-D. Knuth, "The Art of Computer Programming" (vol. 2, Seminumerical Algorithms) (3rd ed.) [1997-11-14].

-R. P. Brent and P. Zimmermann, “Modern computer arithmetic”, Cambridge Monographs on Applied and Computational Mathematics, vol. 18, Cambridge University Press, Cambridge, 2011. MR 2760886

-Homepage of David Harvey; https://web.maths.unsw.edu.au/~davidharvey/

-Article by D. Harvey: https://theconversation.com/weve-found-a-quicker-way-to-multiply-really-big-numbers-114923

-(And some wikipedia…)