On the computational complexity of mathematical functions

Jean-Pierre Demailly

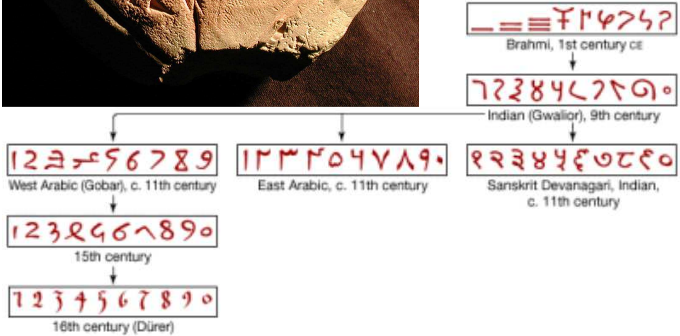
Institut Fourier, Université de Grenoble I
& Académie des Sciences, Paris (France)

November 26, 2011
KVPY conference at Vijyoshi Camp

Computing, a very old concern

Babylonian mathematical tablet allowing the computation of $\sqrt{2}$ (1800 – 1600 BC)

Decimal numeral system invented in India ($\sim$ 500BC ?) :



Madhava's formula for π

Early calculations of π were done by Greek and Indian mathematicians several centuries BC.

Madhava's formula for π

Early calculations of π were done by Greek and Indian mathematicians several centuries BC.
These early evaluations used polygon approximations and Pythagoras theorem. In this way, using 96 sides, Archimedes got $3 + \frac{10}{71} < \pi < 3 + \frac{10}{70}$ whose average is 3.1418 (c. 230 BC).
Chinese mathematicians reached 7 decimal places in 480 AD.

Madhava's formula for π

Early calculations of π were done by Greek and Indian mathematicians several centuries BC.
These early evaluations used polygon approximations and Pythagoras theorem. In this way, using 96 sides, Archimedes got $3 + \frac{10}{71} < \pi < 3 + \frac{10}{70}$ whose average is 3.1418 (c. 230 BC).
Chinese mathematicians reached 7 decimal places in 480 AD.

The next progress was the discovery of the first infinite series formula by Madhava (circa 1350 – 1450), a prominent mathematician-astronomer from Kerala (formula rediscovered in the XVIIe century by Leibniz and Gregory) :

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots + \frac{(-1)^n}{2n+1} + \cdots$$

Convergence is unfortunately very slow, but Madhava was able to improve convergence and reached in this way 11 decimal places.

Ramanujan's formula for π

Srinivasa Ramanujan (1887 – 1920), a self-taught mathematical prodigee. His work dealt mainly with arithmetics and function theory

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{9801} \sum_{n=0}^{+\infty} \frac{(4n)!(1103 + 26390n)}{(n!)^4 396^{4n}} \qquad (1910).$$

Ramanujan's formula for π

Srinivasa Ramanujan (1887 – 1920), a self-taught mathematical prodigee. His work dealt mainly with arithmetics and function theory

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{9801} \sum_{n=0}^{+\infty} \frac{(4n)!(1103 + 26390n)}{(n!)^4 396^{4n}} \qquad (1910).$$

Each term is approximately 10^8 times smaller than the preceding one, so the convergence is very fast.

Computational complexity theory

- Complexity theory is a branch of computer science and mathematics that :
  – tries to classify problems according to their difficulty
  – focuses on the number of steps (or time) needed to solve them.

Computational complexity theory

- Complexity theory is a branch of computer science and mathematics that :
  – tries to classify problems according to their difficulty
  – focuses on the number of steps (or time) needed to solve them.
- Let $N =$ size of the data (e.g. for a decimal number, the number N of digits.)

A problem will be said to have polynomial complexity if it requires less than $C\,N^d$ steps (or units of time) to be solved, where C and d are constants (d is the degree).

Computational complexity theory

- Complexity theory is a branch of computer science and mathematics that :

– tries to classify problems according to their difficulty

– focuses on the number of steps (or time) needed to solve them.

- Let N = size of the data (e.g. for a decimal number, the number N of digits.)

A problem will be said to have polynomial complexity if it requires less than $C\,N^d$ steps (or units of time) to be solved, where C and d are constants (d is the degree).

- Especially, it is said to have

– linear complexity when $\#\,\text{steps} \le C\,N$

Computational complexity theory

- Complexity theory is a branch of computer science and mathematics that :

– tries to classify problems according to their difficulty

– focuses on the number of steps (or time) needed to solve them.

- Let N = size of the data (e.g. for a decimal number, the number N of digits.)

A problem will be said to have polynomial complexity if it requires less than $C\,N^d$ steps (or units of time) to be solved, where C and d are constants (d is the degree).

- Especially, it is said to have

– linear complexity when $\#\,\text{steps} \le C\,N$

– quadratic complexity when $\#\,\text{steps} \le C\,N^2$

Computational complexity theory

- Complexity theory is a branch of computer science and mathematics that :

– tries to classify problems according to their difficulty

– focuses on the number of steps (or time) needed to solve them.

- Let $N =$ size of the data (e.g. for a decimal number, the number N of digits.)

A problem will be said to have polynomial complexity if it requires less than $C\,N^d$ steps (or units of time) to be solved, where C and d are constants (d is the degree).

- Especially, it is said to have

– linear complexity when $\#\,\text{steps} \leq C\,N$

– quadratic complexity when $\#\,\text{steps} \leq C\,N^2$

– quasi-linear complexity when $\#\,\text{steps} \leq C_\varepsilon\,N^{1+\varepsilon}$, $\forall \varepsilon > 0$.

First observations about complexity

- Addition has linear complexity:
consider decimal numbers of the form $0.a_1a_2a_3\ldots a_N$, $0.b_1b_2b_3\ldots b_N$, we have

$$\sum_{1\le n\le N} a_n 10^{-n} + \sum_{1\le n\le N} b_n 10^{-n} = \sum_{1\le n\le N} (a_n+b_n) 10^{-n},$$

taking carries into account, this is done in N steps at most.

First observations about complexity

- Addition has linear complexity:
consider decimal numbers of the form $0.a_1 a_2 a_3 \ldots a_N$, $0.b_1 b_2 b_3 \ldots b_N$, we have

$$\sum_{1 \le n \le N} a_n 10^{-n} + \sum_{1 \le n \le N} b_n 10^{-n} = \sum_{1 \le n \le N} (a_n + b_n) 10^{-n},$$

taking carries into account, this is done in N steps at most.

- What about multiplication ?

First observations about complexity

- Addition has linear complexity:

consider decimal numbers of the form $0.a_1a_2a_3\ldots a_N$, $0.b_1b_2b_3\ldots b_N$, we have

$$\sum_{1\le n\le N} a_n 10^{-n} + \sum_{1\le n\le N} b_n 10^{-n} = \sum_{1\le n\le N} (a_n+b_n)10^{-n},$$

taking carries into account, this is done in N steps at most.

- What about multiplication ?

$$\sum_{1\le k\le N} a_k 10^{-k} \times \sum_{1\le \ell\le N} b_\ell 10^{-\ell} = \sum_{1\le n\le N} c_n 10^{-n}, \quad c_n = \sum_{k+\ell=n} a_k b_\ell.$$

Calculation of each c_n requires at most N elementary multiplications and $N-1$ additions and corresponding carries, thus the algorithm requires less than $N\times 3N$ steps.

Thus multiplication has at most quadratic complexity.

The Karatsuba algorithm

Can one do better than quadratic complexity for multiplication?

The Karatsuba algorithm

Can one do better than quadratic complexity for multiplication?

Yes !! It was discovered by Karatsuba around 1960 that multiplication has complexity less than $C\,N^{\log_2 3} \simeq C\,N^{1.585}$

Karatsuba's idea: for $N = 2q$ even, split $x = 0.a_1a_2\ldots a_N$ as

$$x = x' + 10^{-q}x'', \quad x' = 0.a_1a_2\ldots a_q, \quad x'' = 0.a_{q+1}a_{q+2}\ldots a_{2q}$$

and similarly $y = 0.b_1b_2\ldots b_N = y' + 10^{-q}y''$. To calculate xy, one would normally need $x'y'$, $x''y''$ and $x'y'' + x''y'$ which take 4 multiplications and 1 addition of q-digit numbers. However, one can use only 3 multiplications by calculating

$$x'y', \quad x''y'', \quad x'y'' + x''y' = x'y' + x''y'' - (x' - x'')(y' - y'')$$

(at the expense of 4 additions).

The Karatsuba algorithm

Can one do better than quadratic complexity for multiplication?

Yes !! It was discovered by Karatsuba around 1960 that multiplication has complexity less than $C\,N^{\log_2 3} \simeq C\,N^{1.585}$

Karatsuba's idea: for $N = 2q$ even, split $x = 0.a_1 a_2 \ldots a_N$ as

$$x = x' + 10^{-q} x'', \quad x' = 0.a_1 a_2 \ldots a_q, \quad x'' = 0.a_{q+1} a_{q+2} \ldots a_{2q}$$

and similarly $y = 0.b_1 b_2 \ldots b_N = y' + 10^{-q} y''$. To calculate xy, one would normally need $x'y'$, $x''y''$ and $x'y'' + x''y'$ which take 4 multiplications and 1 addition of q-digit numbers. However, one can use only 3 multiplications by calculating

$$x'y', \quad x''y'', \quad x'y'' + x''y' = x'y' + x''y'' - (x' - x'')(y' - y'')$$

(at the expense of 4 additions). One then proceeds inductively to conclude that the time $T(N)$ needed for $N = 2^s$ satisfies

$$T(2^s) \leq 3\,T(2^{s-1}) + 4\,2^{s-1}.$$

Optimal complexity of multiplication

It is an easy exercise to conclude by induction that $T(2^s) \leq 6\,3^s - 4\,2^s$ if one assumes $T(1) = 1$, and so

$$T(2^s) \leq 6\,3^s \quad \Rightarrow \quad T(N) \leq C\,N^{\log_2 3}.$$

Optimal complexity of multiplication

It is an easy exercise to conclude by induction that $T(2^s) \le 6\,3^s - 4\,2^s$ if one assumes $T(1) = 1$, and so

$$T(2^s) \le 6\,3^s \quad \Rightarrow \quad T(N) \le C\,N^{\log_2 3}.$$

It was in fact shown in 1971 by Schönage and Strassen that multiplication has quasi-linear complexity, less than

$$C\,N\,\log N\,\log\log N.$$

Optimal complexity of multiplication

It is an easy exercise to conclude by induction that $T(2^s) \leq 6\,3^s - 4\,2^s$ if one assumes $T(1) = 1$, and so

$$T(2^s) \leq 6\,3^s \quad \Rightarrow \quad T(N) \leq C\,N^{\log_2 3}.$$

It was in fact shown in 1971 by Schönage and Strassen that multiplication has quasi-linear complexity, less than

$$C\,N\,\log N\,\log\log N.$$

For this reason, the usual mathematical functions also have quasi-linear complexity at most !

Optimal complexity of multiplication

It is an easy exercise to conclude by induction that $T(2^s) \leq 6\,3^s - 4\,2^s$ if one assumes $T(1) = 1$, and so

$$T(2^s) \leq 6\,3^s \quad \Rightarrow \quad T(N) \leq C\,N^{\log_2 3}.$$

It was in fact shown in 1971 by Schönage and Strassen that multiplication has quasi-linear complexity, less than

$$C\,N\,\log N\,\log\log N.$$

For this reason, the usual mathematical functions also have quasi-linear complexity at most !

The Schönage-Strassen algorithm is based on the use of discrete Fourier transforms. The theory comes from Joseph Fourier, the founder of my university in 1810 ...

Joseph Fourier

Joseph Fourier (1768 – 1830)
in his suit of member of
Académie des Sciences,
of which he became
"Secrétaire Perpétuel"
(Head) in 1822.

Life of Joseph Fourier

Born in 1768 in a poor family, Joseph Fourier quickly reveals himself to be a scientific prodigee.

Life of Joseph Fourier

Born in 1768 in a poor family, Joseph Fourier quickly reveals himself to be a scientific prodigee.

Orphan from mother at age 8 and from father at age 10, he is sent to a religious military school in the city of Auxerre, where he has fortunately access to some important scientific books.

Life of Joseph Fourier

Born in 1768 in a poor family, Joseph Fourier quickly reveals himself to be a scientific prodigee.

Orphan from mother at age 8 and from father at age 10, he is sent to a religious military school in the city of Auxerre, where he has fortunately access to some important scientific books.

He is just $16\frac{1}{2}$ years when the director of his school asks him to become the math teacher !

Life of Joseph Fourier

Born in 1768 in a poor family, Joseph Fourier quickly reveals himself to be a scientific prodigee.

Orphan from mother at age 8 and from father at age 10, he is sent to a religious military school in the city of Auxerre, where he has fortunately access to some important scientific books.

He is just $16\frac{1}{2}$ years when the director of his school asks him to become the math teacher !

At age 26, he becomes a Professor at Ecole Normale Supérieure and École Polytechnique. In 1798, he is chosen by Napoleon as his main scientific advisor during the campaign of Egypt.

Life of Joseph Fourier

Born in 1768 in a poor family, Joseph Fourier quickly reveals himself to be a scientific prodigee.

Orphan from mother at age 8 and from father at age 10, he is sent to a religious military school in the city of Auxerre, where he has fortunately access to some important scientific books.

He is just $16\frac{1}{2}$ years when the director of his school asks him to become the math teacher !

At age 26, he becomes a Professor at Ecole Normale Supérieure and École Polytechnique. In 1798, he is chosen by Napoleon as his main scientific advisor during the campaign of Egypt.

Back in France in 1802, he becomes the Governor of the Grenoble area and founds the University. During this period, he discovers the heat equation and what is now called Fourier analysis...

Life of Joseph Fourier

Born in 1768 in a poor family, Joseph Fourier quickly reveals himself to be a scientific prodigee.

Orphan from mother at age 8 and from father at age 10, he is sent to a religious military school in the city of Auxerre, where he has fortunately access to some important scientific books.

He is just $16\frac{1}{2}$ years when the director of his school asks him to become the math teacher !

At age 26, he becomes a Professor at Ecole Normale Supérieure and École Polytechnique. In 1798, he is chosen by Napoleon as his main scientific advisor during the campaign of Egypt.

Back in France in 1802, he becomes the Governor of the Grenoble area and founds the University. During this period, he discovers the heat equation and what is now called Fourier analysis...

In 1824, he predicts the green house effect !

Heat equation and Fourier series

Let $\theta(x, y, z, t)$ be the the temperature of a physical material at a point (x, y, z) and at time t.

Heat equation and Fourier series

Let $\theta(x, y, z, t)$ be the the temperature of a physical material at a point (x, y, z) and at time t.

Fourier shows theoretically and experimentally around 1807 that $\theta(x, y, z, t)$ satisfies the propagation equation

$$\theta'_t = D(\theta''_{xx} + \theta''_{yy} + \theta''_{zz}).$$

where D is a constant characterizing the material.

Heat equation and Fourier series

Let $\theta(x, y, z, t)$ be the the temperature of a physical material at a point (x, y, z) and at time t.

Fourier shows theoretically and experimentally around 1807 that $\theta(x, y, z, t)$ satisfies the propagation equation

$$\theta'_t = D(\theta''_{xx} + \theta''_{yy} + \theta''_{zz}).$$

where D is a constant characterizing the material.

He then shows that in many cases the solutions can be expressed in terms of trigonometric series

$$f(x) = \sum_{n=0}^{+\infty} a_n \cos n\omega x + b_n \sin n\omega x = \sum_{n=-\infty}^{+\infty} c_n e^{in\omega x}$$

Heat equation and Fourier series

Let $\theta(x, y, z, t)$ be the the temperature of a physical material at a point (x, y, z) and at time t.

Fourier shows theoretically and experimentally around 1807 that $\theta(x, y, z, t)$ satisfies the propagation equation

$$\theta'_t = D(\theta''_{xx} + \theta''_{yy} + \theta''_{zz}).$$

where D is a constant characterizing the material.

He then shows that in many cases the solutions can be expressed in terms of trigonometric series

$$f(x) = \sum_{n=0}^{+\infty} a_n \cos n\omega x + b_n \sin n\omega x = \sum_{n=-\infty}^{+\infty} c_n e^{in\omega x}$$

In fact all periodic phenomena can be described in this way. This is the basis of the modern theory of signal processing and electromagnetism.

Discrete Fourier transform

Let $(a_n)_{0\le n<N}$ be a finite sequence of numbers and let u be a primitive N-th root of unity, i.e.

$$u^N = 1 \quad \text{but} \quad u^n \neq 1 \ \text{ for } 0 < n < N.$$

Discrete Fourier transform

Let $(a_n)_{0\le n<N}$ be a finite sequence of numbers and let u be a primitive N-th root of unity, i.e.

$$u^N = 1 \quad \text{but} \quad u^n \neq 1 \ \text{ for } 0 < n < N.$$

One can work with complex numbers and take $u = e^{2\pi i/N}$.

Discrete Fourier transform

Let $(a_n)_{0\le n<N}$ be a finite sequence of numbers and let u be a primitive N-th root of unity, i.e.

$$u^N = 1 \quad \text{but} \quad u^n \neq 1 \ \text{ for } 0 < n < N.$$

One can work with complex numbers and take $u = e^{2\pi i/N}$.

When working with integers, it is easier to work modulo a large prime number, e.g. $p = 65537$ and take $N = p - 1 = 65536$. Then $u = 3$ satisfies $u^N = 1 \bmod p$ and one can check that $u = 3$ is a primitive N-root of unity.

Discrete Fourier transform

Let $(a_n)_{0\le n<N}$ be a finite sequence of numbers and let u be a primitive N-th root of unity, i.e.

$$u^N = 1 \quad \text{but} \quad u^n \neq 1 \ \text{ for } 0 < n < N.$$

One can work with complex numbers and take $u = e^{2\pi i/N}$.

When working with integers, it is easier to work modulo a large prime number, e.g. $p = 65537$ and take $N = p - 1 = 65536$. Then $u = 3$ satisfies $u^N = 1 \bmod p$ and one can check that $u = 3$ is a primitive N-root of unity.

The discrete Fourier transform of (a_n) is the sequence

$$\widehat{a}_n = \sum_{k=0}^{N-1} a_k u^{kn}.$$

Discrete Fourier transform

Let $(a_n)_{0 \le n < N}$ be a finite sequence of numbers and let u be a primitive N-th root of unity, i.e.

$$u^N = 1 \quad \text{but} \quad u^n \neq 1 \ \text{ for } 0 < n < N.$$

One can work with complex numbers and take $u = e^{2\pi i/N}$.

When working with integers, it is easier to work modulo a large prime number, e.g. $p = 65537$ and take $N = p - 1 = 65536$. Then $u = 3$ satisfies $u^N = 1 \bmod p$ and one can check that $u = 3$ is a primitive N-root of unity.

The discrete Fourier transform of (a_n) is the sequence

$$\widehat{a}_n = \sum_{k=0}^{N-1} a_k u^{kn}.$$

It is convenient to consider that the index n is defined mod N (e.g. a_{-n} means a_{N-n} for $0 < n < N$).

Discrete Fourier transform

Let $(a_n)_{0\le n<N}$ be a finite sequence of numbers and let u be a primitive N-th root of unity, i.e.

$$u^N = 1 \quad \text{but} \quad u^n \neq 1 \ \text{ for } 0 < n < N.$$

One can work with complex numbers and take $u = e^{2\pi i/N}$.

When working with integers, it is easier to work modulo a large prime number, e.g. $p = 65537$ and take $N = p - 1 = 65536$. Then $u = 3$ satisfies $u^N = 1 \bmod p$ and one can check that $u = 3$ is a primitive N-root of unity.

The discrete Fourier transform of (a_n) is the sequence

$$\widehat{a}_n = \sum_{k=0}^{N-1} a_k u^{kn}.$$

It is convenient to consider that the index n is defined mod N (e.g. a_{-n} means a_{N-n} for $0 < n < N$).

Main formulas of Fourier theory

Fourier transform of a convolution:

For $a = (a_n)$ and $b = (b_n)$ define $c = a * b$ to be the sequence

$$c_n = \sum_{p+q=n \mod N} a_p b_q \quad \text{“convolution of } a \text{ and } b.\text{”}$$

Then $\widehat{c}_n = \widehat{a}_n \widehat{b}_n$.

Main formulas of Fourier theory

Fourier transform of a convolution:

For $a = (a_n)$ and $b = (b_n)$ define $c = a * b$ to be the sequence

$$c_n = \sum_{p+q=n \mod N} a_p b_q \quad \text{“convolution of } a \text{ and } b.\text{”}$$

Then $\widehat{c}_n = \widehat{a}_n \widehat{b}_n$.

Proof. $\displaystyle \sum_s c_s u^{sn} = \sum_s \Big(\sum_{k+\ell=s} a_k b_\ell \Big) u^{sn} = \sum_{k,\ell} a_k u^{kn} b_\ell u^{\ell n} = \widehat{a}_n \widehat{b}_n.$

Main formulas of Fourier theory

Fourier transform of a convolution:

For $a = (a_n)$ and $b = (b_n)$ define $c = a * b$ to be the sequence

$$c_n = \sum_{p+q=n \mod N} a_p b_q \quad \text{“convolution of } a \text{ and } b\text{.”}$$

Then $\widehat{c}_n = \widehat{a}_n \widehat{b}_n$.

Proof. $\sum_s c_s u^{sn} = \sum_s \Big(\sum_{k+\ell=s} a_k b_\ell \Big) u^{sn} = \sum_{k,\ell} a_k u^{kn} b_\ell u^{\ell n} = \widehat{a}_n \widehat{b}_n$.

Fourier inversion formula: applying twice the Fourier transform, one gets

$$\widehat{\widehat{a}}_n = N\, a_{-n} = -a_{-n} \mod p \qquad (\text{recall } N = p-1).$$

Main formulas of Fourier theory

Fourier transform of a convolution:
For $a = (a_n)$ and $b = (b_n)$ define $c = a * b$ to be the sequence

$$c_n = \sum_{p+q=n \mod N} a_p b_q \quad \text{“convolution of } a \text{ and } b\text{.”}$$

Then $\widehat{c}_n = \widehat{a}_n \widehat{b}_n$.

Proof. $\sum_s c_s u^{sn} = \sum_s \Big(\sum_{k+\ell=s} a_k b_\ell \Big) u^{sn} = \sum_{k,\ell} a_k u^{kn} b_\ell u^{\ell n} = \widehat{a}_n \widehat{b}_n$.

Fourier inversion formula: applying twice the Fourier transform, one gets

$$\widehat{\widehat{a}}_n = N\, a_{-n} = -a_{-n} \mod p \qquad (\text{recall } N = p-1).$$

Proof. $\widehat{\widehat{a}}_n = \sum_k \Big(\sum_\ell a_\ell u^{k\ell} \Big) u^{kn} = \sum_\ell a_\ell \Big(\sum_k u^{k(n+\ell)} \Big)$ and $\sum_k u^{k(n+\ell)} = 0$ if $\ell \neq -n$ and $\sum_k u^{k(n+\ell)} = N$ if $\ell = -n$.

Fast Fourier Transform (FFT)

Consequence: To calculate the convolution $c = a * b$ (which is what we need to calculate $\sum a_k 10^{-k} \sum b_\ell 10^{-\ell}$), one calculates the Fourier transforms $(\widehat{a}_n)$, $(\widehat{b}_n)$, then $\widehat{c}_n = \widehat{a}_n \widehat{b}_n$, which gives back $(-c_{-n})$ and thus (c_n) by Fourier inversion.

Fast Fourier Transform (FFT)

Consequence: To calculate the convolution $c = a * b$ (which is what we need to calculate $\sum a_k 10^{-k} \sum b_\ell 10^{-\ell}$), one calculates the Fourier transforms $(\widehat{a}_n)$, $(\widehat{b}_n)$, then $\widehat{c}_n = \widehat{a}_n \widehat{b}_n$, which gives back $(-c_{-n})$ and thus (c_n) by Fourier inversion.

This looks complicated, but the Fourier transform can be computed extremely fast !!

FFT algorithm: assume that $N = 2^s$ (in our example $N = 65536 = 2^{16}$) and define inductively $\alpha_{n,0} = a_n$ and

$$\alpha_{n,k+1} = \alpha_{n,k} + \alpha_{n+2^k} u^{2^k n}, \quad 0 \le k < s.$$

Fast Fourier Transform (FFT)

Consequence: To calculate the convolution $c = a * b$ (which is what we need to calculate $\sum a_k 10^{-k} \sum b_\ell 10^{-\ell}$), one calculates the Fourier transforms $(\widehat{a}_n)$, $(\widehat{b}_n)$, then $\widehat{c}_n = \widehat{a}_n \widehat{b}_n$, which gives back $(-c_{-n})$ and thus (c_n) by Fourier inversion.

This looks complicated, but the Fourier transform can be computed extremely fast !!

FFT algorithm: assume that $N = 2^s$ (in our example $N = 65536 = 2^{16}$) and define inductively $\alpha_{n,0} = a_n$ and

$$\alpha_{n,k+1} = \alpha_{n,k} + \alpha_{n+2^k} u^{2^k n}, \quad 0 \le k < s.$$

By considering the binary decomposition $n = \sum n_k 2^k$, $0 \le k < s$, of any integer $n = 0 \ldots N-1$, one sees that $\alpha_{n,s} = \widehat{a}_n$. The calculation requires only s steps, each of which requires N additions and $2N$ mutiplications (using $u^{2^{k+1} n} = (u^{2^k n})^2$), so in total we consume only $3sN = 3N \log_2 N$ operations !

Other mathematical functions

OK about multiplication, but what for division ? square root ?

Other mathematical functions

OK about multiplication, but what for division ? square root ?

Approximate division can be obtained solely from multiplication! If x_0 is a rough approximation of $1/a$, then the sequence

$$x_{n+1} = 2x_n - ax_n^2$$

satisfies $1 - ax_{n+1} = (1 - ax_n)^2$, and so inductively $1 - ax_n = (1 - ax_0)^{2^n}$ will converge extremely fast to 0. In fact if $|1 - ax_0| < 1/10$ and $n \sim \log_2 N$, we get already N correct digits. Hence we need iterating only $\log_2 N$ times the sequence, and so division is also quasi-linear in time.

Other mathematical functions

OK about multiplication, but what for division ? square root ?

Approximate division can be obtained solely from multiplication! If x_0 is a rough approximation of $1/a$, then the sequence

$$x_{n+1} = 2x_n - ax_n^2$$

satisfies $1 - ax_{n+1} = (1 - ax_n)^2$, and so inductively $1 - ax_n = (1 - ax_0)^{2^n}$ will converge extremely fast to 0. In fact if $|1 - ax_0| < 1/10$ and $n \sim \log_2 N$, we get already N correct digits. Hence we need iterating only $\log_2 N$ times the sequence, and so division is also quasi-linear in time.

Similarly, square roots can be approximated by using only multiplications and divisions, thanks to the "Babylonian algorithm":

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right), \qquad x_0 > 0$$

What about π ?

In fact Carl-Friedrich Gauss (another mathematical prodigee...) discovered around 1797 the following formula for the arithmetic-geometric mean:
start from real numbers $a, b > 0$ and define inductively $a_0 = a$, $b_0 = b$ and

$$a_{n+1} = \frac{a_n + b_n}{2}, \qquad b_{n+1} = \sqrt{a_n b_n}.$$

What about π ?

In fact Carl-Friedrich Gauss (another mathematical prodigee...) discovered around 1797 the following formula for the arithmetic-geometric mean:
start from real numbers $a, b > 0$ and define inductively $a_0 = a$, $b_0 = b$ and

$$a_{n+1} = \frac{a_n + b_n}{2}, \qquad b_{n+1} = \sqrt{a_n b_n}.$$

Then (a_n) and (b_n) converge (extremely fast, only $\sim \log_2 N$ steps to get N correct digits) towards

$$M(a,b) = \frac{2\pi}{I(a,b)} \quad \text{where } I(a,b) = \int_0^{2\pi} \frac{dx}{\sqrt{a^2\cos^2 x + b^2 \sin^2 x}}$$

(an "elliptic integral").

The Brent-Salamin formula

Using this and another formula due to Legendre (1752 – 1833), Brent and Salamin found in 1976 a remarkable formula for π. Define

$$c_n = \sqrt{a_n^2 - b_n^2}$$

in the arithmetic-geometric sequence. Then

$$\pi = \frac{4\,M(1, 1/\sqrt{2})^2}{1 - \sum_{n=1}^{+\infty} 2^{n+1} c_n^2}.$$

The Brent-Salamin formula

Using this and another formula due to Legendre (1752 – 1833), Brent and Salamin found in 1976 a remarkable formula for π. Define

$$c_n = \sqrt{a_n^2 - b_n^2}$$

in the arithmetic-geometric sequence. Then

$$\pi = \frac{4\,M(1, 1/\sqrt{2})^2}{1 - \sum_{n=1}^{+\infty} 2^{n+1} c_n^2}.$$

As a consequence, the calculation of N digits of π is also a quasi-linear problem!

The Brent-Salamin formula

Using this and another formula due to Legendre (1752 – 1833), Brent and Salamin found in 1976 a remarkable formula for π. Define

$$c_n = \sqrt{a_n^2 - b_n^2}$$

in the arithmetic-geometric sequence. Then

$$\pi = \frac{4\, M(1, 1/\sqrt{2})^2}{1 - \sum_{n=1}^{+\infty} 2^{n+1} c_n^2}.$$

As a consequence, the calculation of N digits of π is also a quasi-linear problem!

This formula has been used several times to break the world record, which seems to be 5 trillions digits since 2010 (however, there exist so efficient quadratic complexity formulas that they are still competitive at that level...)

Complexity of matrix multiplication

Question. How many steps are necessary to compute the product $C = AB$ of two $n \times n$ matrices, assuming that each elementary multiplication or addition takes 1 step?

Complexity of matrix multiplication

Question. How many steps are necessary to compute the product $C = AB$ of two $n \times n$ matrices, assuming that each elementary multiplication or addition takes 1 step?

The standard matrix matrix multiplication algorithm

$$c_{ik} = \sum_{1 \le j \le n} a_{ij} b_{jk}, \qquad 1 \le i, k \le n$$

leads to calculate n^2 coefficients, each of which requires n multiplications and $(n-1)$ additions, so in total $n^2(2n-1) \sim 2n^3$ operations.

Complexity of matrix multiplication

Question. How many steps are necessary to compute the product $C = AB$ of two $n \times n$ matrices, assuming that each elementary multiplication or addition takes 1 step?

The standard matrix matrix multiplication algorithm

$$c_{ik} = \sum_{1 \le j \le n} a_{ij} b_{jk}, \qquad 1 \le i, k \le n$$

leads to calculate n^2 coefficients, each of which requires n multiplications and $(n-1)$ additions, so in total $n^2(2n-1) \sim 2n^3$ operations.

However, the size of the data is $N = n^2$, and the general philosophy that it should be quasi-linear would suggest an algorithm with complexity less than $N^{1+\varepsilon} = n^{2+2\varepsilon}$ for every ε.

Complexity of matrix multiplication

Question. How many steps are necessary to compute the product $C = AB$ of two $n \times n$ matrices, assuming that each elementary multiplication or addition takes 1 step?

The standard matrix matrix multiplication algorithm

$$c_{ik} = \sum_{1 \le j \le n} a_{ij} b_{jk}, \qquad 1 \le i, k \le n$$

leads to calculate n^2 coefficients, each of which requires n multiplications and $(n-1)$ additions, so in total $n^2(2n-1) \sim 2n^3$ operations.

However, the size of the data is $N = n^2$, and the general philosophy that it should be quasi-linear would suggest an algorithm with complexity less than $N^{1+\varepsilon} = n^{2+2\varepsilon}$ for every ε.

The fastest known algorithm, due to Coppersmith and Winograd in 1987 has $\#\text{steps} \le C\, n^{2.38}$ (quite complicated!)