N-th term of a linear recurrent sequence

Alin Bostan

Inria informatics mathematics

MPRI

C-2-22

October 25, 2022

Faster Matrix Multiplication via Asymmetric Hashing

Ran Duan *
Tsinghua University

Hongxun Wu †
UC Berkeley

Renfei Zhou ‡
Tsinghua University

October 20, 2022

Abstract

Fast matrix multiplication is one of the most fundamental problems in algorithm research. The exponent of the optimal time complexity of matrix multiplication is usually denoted by ω. This paper discusses new ideas for improving the laser method for fast matrix multiplication. We observe that the analysis of higher powers of the Coppersmith-Winograd tensor [Coppersmith & Winograd 1990] incurs a "combination loss", and we partially compensate it by using an asymmetric version of CW's hashing method. By analyzing the 8th power of the CW tensor, we give a new bound of $\omega < 2.37188$, which improves the previous best bound of $\omega < 2.37286$ [Alman & V.Williams 2020]. Our result breaks the lower bound of 2.3725 in [Ambainis et al. 2014] because of the new method for analyzing component (constituent) tensors.

▷ Improves previous best upper bound $\omega < 2.37286$ to $\omega < 2.37188$.

The exercise from last week

Prove the identity

$$\arcsin(x)^2 = \sum_{k\geq 0} \frac{k!}{\left(\frac{1}{2}\right)\cdots\left(k+\frac{1}{2}\right)} \frac{x^{2k+2}}{2k+2},$$

by performing the following steps:

1. Show that $y = \arcsin(x)$ can be represented by the differential equation $(1-x^2)y'' - xy' = 0$ and the initial conditions $y(0) = 0$, $y'(0) = 1$.
2. Compute a linear differential equation satisfied by $z(x) = y(x)^2$.
3. Deduce a linear recurrence relation satisfied by the coefficients of $z(x)$.
4. Conclude.

Solution, Part 1.

The starting point is the identity

$$(\arcsin(x))' = \frac{1}{\sqrt{1-x^2}},$$

which allows to represent $\arcsin(x)$ by the differential equation

$$(1-x^2)y'' - xy' = 0$$

together with the initial conditions

$$y(0) = \arcsin(0) = 0, \quad y'(0) = \frac{1}{\sqrt{1-0^2}} = 1.$$

Solution, Part 2.

Let $z = y^2$, with $y'' = \frac{x}{1-x^2} y'$.

Solution, Part 2.

Let $z = y^2$, with $y'' = \frac{x}{1-x^2} y'$. By successive differentiations, we get

Solution, Part 2.

Let $z = y^2$, with $y'' = \frac{x}{1-x^2} y'$. By successive differentiations, we get

$$\begin{aligned}
z' &= 2yy', \\
z'' &= 2y'^2 + 2yy'' = 2y'^2 + \frac{2x}{1-x^2} yy', \\
z''' &= 4y'y'' + \frac{2x}{1-x^2}(y'^2 + yy'') + \left(\frac{2}{1-x^2} + \frac{4x^2}{(1-x^2)^2}\right) yy' \\
&= \left(\frac{2}{1-x^2} + \frac{6x^2}{(1-x^2)^2}\right) yy' + \frac{6x}{1-x^2} y'^2.
\end{aligned}$$

Solution, Part 2.

Let $z = y^2$, with $y'' = \frac{x}{1-x^2} y'$. By successive differentiations, we get

$$
\begin{aligned}
z' &= 2yy', \\
z'' &= 2y'^2 + 2yy'' = 2y'^2 + \frac{2x}{1-x^2} yy', \\
z''' &= 4y'y'' + \frac{2x}{1-x^2}(y'^2 + yy'') + \left(\frac{2}{1-x^2} + \frac{4x^2}{(1-x^2)^2} \right) yy' \\
&= \left(\frac{2}{1-x^2} + \frac{6x^2}{(1-x^2)^2} \right) yy' + \frac{6x}{1-x^2} y'^2.
\end{aligned}
$$

▷ z, z', z'', z''' are $\mathbb{Q}(x)$-linear comb. of y^2, yy', y'^2, thus $\mathbb{Q}(x)$-dependent

Let $z = y^2$, with $y'' = \frac{x}{1-x^2} y'$. By successive differentiations, we get

$$\begin{aligned}
z' &= 2yy', \\
z'' &= 2y'^2 + 2yy'' = 2y'^2 + \frac{2x}{1-x^2} yy', \\
z''' &= 4y'y'' + \frac{2x}{1-x^2}(y'^2 + yy'') + \left(\frac{2}{1-x^2} + \frac{4x^2}{(1-x^2)^2}\right) yy' \\
&= \left(\frac{2}{1-x^2} + \frac{6x^2}{(1-x^2)^2}\right) yy' + \frac{6x}{1-x^2} y'^2.
\end{aligned}$$

▷ z, z', z'', z''' are $\mathbb{Q}(x)$-linear comb. of y^2, yy', y'^2, thus $\mathbb{Q}(x)$-dependent

▷ A dependence relation is determined by computing the kernel of

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & \frac{2x}{1-x^2} & \frac{2}{1-x^2} + \frac{6x^2}{(1-x^2)^2} \\ 0 & 0 & 2 & \frac{6x}{1-x^2} \end{bmatrix}$$

Solution, Part 2.

Let $z = y^2$, with $y'' = \frac{x}{1-x^2} y'$. By successive differentiations, we get

$$\begin{aligned}
z' &= 2yy', \\
z'' &= 2y'^2 + 2yy'' = 2y'^2 + \frac{2x}{1-x^2} yy', \\
z''' &= 4y'y'' + \frac{2x}{1-x^2}(y'^2 + yy'') + \left(\frac{2}{1-x^2} + \frac{4x^2}{(1-x^2)^2}\right) yy' \\
&= \left(\frac{2}{1-x^2} + \frac{6x^2}{(1-x^2)^2}\right) yy' + \frac{6x}{1-x^2} y'^2.
\end{aligned}$$

▷ z, z', z'', z''' are $\mathbb{Q}(x)$-linear comb. of y^2, yy', y'^2, thus $\mathbb{Q}(x)$-dependent

▷ A dependence relation is determined by computing the kernel of

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & \frac{2x}{1-x^2} & \frac{2}{1-x^2} + \frac{6x^2}{(1-x^2)^2} \\ 0 & 0 & 2 & \frac{6x}{1-x^2} \end{bmatrix}$$

▷ The kernel of M is generated by $[0,\ 1,\ 3x,\ x^2-1]^T$

▷ The corresponding differential equation is

$$(x^2-1)z''' + 3xz'' + z' = 0.$$

Solution, Part 2., a variant

Let $z = y^2$, with $y'' = \frac{x}{1-x^2} y'$.

Solution, Part 2., a variant

Let $z = y^2$, with $y'' = \frac{x}{1-x^2} y'$. By successive differentiations, we get

Solution, Part 2., a variant

Let $z = y^2$, with $y'' = \frac{x}{1-x^2} y'$. By successive differentiations, we get

$$\begin{aligned}
z' &= 2yy', \\
z'' &= 2y'^2 + 2yy'' = 2y'^2 + \frac{2x}{1-x^2} yy' = 2y'^2 + \frac{x}{1-x^2} z', \\
z''' &= 4y'y'' + \frac{x}{1-x^2} z'' + \left(\frac{1}{1-x^2} + \frac{2\,x^2}{(1-x^2)^2} \right) z' \\
&= \frac{4x}{1-x^2} y'^2 + \frac{x}{1-x^2} z'' + \frac{x^2+1}{(x^2-1)^2} z' \\
&= \frac{2x}{1-x^2} \left(z'' - \frac{x}{1-x^2} z' \right) + \frac{x}{1-x^2} z'' + \frac{x^2+1}{(x^2-1)^2} z'.
\end{aligned}$$

▷ The corresponding differential equation is

$$(x^2-1)z''' + 3xz'' + z' = 0.$$

Solution, Part 3.

▷ Write $z(x) = \sum_n a_n x^n$. Then:

Solution, Part 3.

▷ Write $z(x) = \sum_n a_n x^n$. Then:

$$z' = \sum_n (n+1) a_{n+1} x^n,$$

Solution, Part 3.

▷ Write $z(x) = \sum_n a_n x^n$. Then:

$$z' = \sum_n (n+1) a_{n+1} x^n,$$

$$z'' = \sum_n (n+1)(n+2) a_{n+2} x^n,$$

Solution, Part 3.

▷ Write $z(x) = \sum_n a_n x^n$. Then:

$$z' = \sum_n (n+1) a_{n+1} x^n,$$

$$z'' = \sum_n (n+1)(n+2) a_{n+2} x^n,$$

$$z''' = \sum_n (n+1)(n+2)(n+3) a_{n+3} x^n.$$

Solution, Part 3.

▷ Write $z(x) = \sum_n a_n x^n$. Then:

$$z' = \sum_n (n+1) a_{n+1} x^n,$$

$$z'' = \sum_n (n+1)(n+2) a_{n+2} x^n,$$

$$z''' = \sum_n (n+1)(n+2)(n+3) a_{n+3} x^n.$$

▷ The coefficient of x^n in $(x^2-1)z''' + 3xz'' + z'$ is

$$(n-1)n(n+1)a_{n+1} - (n+1)(n+2)(n+3)a_{n+3} + 3n(n+1)a_{n+1} + (n+1)a_{n+1}$$

Solution, Part 3.

- Write $z(x) = \sum_n a_n x^n$. Then:

$$z' = \sum_n (n+1) a_{n+1} x^n,$$

$$z'' = \sum_n (n+1)(n+2) a_{n+2} x^n,$$

$$z''' = \sum_n (n+1)(n+2)(n+3) a_{n+3} x^n.$$

- The coefficient of x^n in $(x^2-1)z''' + 3xz'' + z'$ is

$$(n-1)n(n+1)a_{n+1} - (n+1)(n+2)(n+3)a_{n+3} + 3n(n+1)a_{n+1} + (n+1)a_{n+1}$$

- Thus, the recurrence corresponding to $(x^2-1)z''' + 3xz'' + z' = 0$ is

$$(n+1)(n+2)(n+3)a_{n+3} = (n+1)^3 a_{n+1}.$$

Solution, Part 3.

- Write $z(x) = \sum_n a_n x^n$. Then:

$$z' = \sum_n (n+1) a_{n+1} x^n,$$

$$z'' = \sum_n (n+1)(n+2) a_{n+2} x^n,$$

$$z''' = \sum_n (n+1)(n+2)(n+3) a_{n+3} x^n.$$

- The coefficient of x^n in $(x^2-1)z''' + 3xz'' + z'$ is

$$(n-1)n(n+1)a_{n+1} - (n+1)(n+2)(n+3)a_{n+3} + 3n(n+1)a_{n+1} + (n+1)a_{n+1}$$

- Thus, the recurrence corresponding to $(x^2-1)z''' + 3xz'' + z' = 0$ is

$$(n+1)(n+2)(n+3)a_{n+3} = (n+1)^3 a_{n+1}.$$

- Since $(n+1)$ has no roots in $\mathbb{N}$, it further simplifies to

$$(n+2)(n+3)a_{n+3} - (n+1)^2 a_{n+1} = 0.$$

Solution, Part 4.

▷ $z = \sum_n a_n x^n$ satisfies

$$(n+2)(n+3)a_{n+3} - (n+1)^2 a_{n+1} = 0.$$

▷ Initial conditions:

$$a_0 = z(0) = y(0)^2 = 0,\ a_1 = z'(0) = 2y(0)y'(0) = 0,\ a_2 = \frac{1}{2}z''(0) = y'(0)^2 = 1.$$

▷ Recurrence and $a_1 = 0$ imply $a_{2k+1} = 0$, so the series is even.

▷ Let $b_k = a_{2k+2}$. Then $z(x) = \sum_k b_k x^{2k+2}$ and

$$(2k+1)(2k+2)b_k = 4k^2 b_{k-1}, \quad b_0 = 1$$

▷ Thus, the sequence $(b_k)_k$ is hypergeometric and

$$b_k = 2\frac{k^2}{(k+1)(2k+1)}b_{k-1} = \cdots = 2^k \frac{k!^2}{(k+1)!(2k+1)(2k-1)\cdots 3}$$

Solution, Part 4.

▷ $z = \sum_n a_n x^n$ satisfies

$$(n+2)(n+3)a_{n+3} - (n+1)^2 a_{n+1} = 0.$$

▷ Initial conditions:

$$a_0 = z(0) = y(0)^2 = 0,\ a_1 = z'(0) = 2y(0)y'(0) = 0,\ a_2 = \frac{1}{2}z''(0) = y'(0)^2 = 1.$$

▷ Recurrence and $a_1 = 0$ imply $a_{2k+1} = 0$, so the series is even.

▷ Let $b_k = a_{2k+2}$. Then $z(x) = \sum_k b_k x^{2k+2}$ and

$$(2k+1)(2k+2)b_k = 4k^2 b_{k-1}, \quad b_0 = 1$$

▷ Thus, the sequence $(b_k)_k$ is hypergeometric and

$$b_k = \frac{k!}{(k+1)\cdot(k+\frac{1}{2})(k-\frac{1}{2})\cdots\frac{3}{2}} = \frac{k!}{(k+\frac{1}{2})(k-\frac{1}{2})\cdots\frac{1}{2}}\,\frac{1}{2k+2} \qquad \square$$

COMPUTING TERMS OF RECURRENT SEQUENCES

Main question

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

Main question

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

▷ Input $(u_n)_{n\geq 0}$ is assumed to be a recurrent sequence, and it is specified by a recurrence relation and enough initial terms

Main question

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

▷ Input $(u_n)_{n\geq 0}$ is assumed to be a recurrent sequence, and it is specified by a recurrence relation and enough initial terms

▷ Efficiency is measured in terms of ring operations, or of bit operations

Main question

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

▷ Input $(u_n)_{n\geq 0}$ is assumed to be a recurrent sequence, and it is specified by a recurrence relation and enough initial terms

▷ Efficiency is measured in terms of ring operations, or of bit operations

---

Two variants:

Given $(u_n)_n$ in $R^{\mathbb{N}}$ and $(N_1, \ldots, N_s) \in \mathbb{N}^s$, compute $(u_{N_1}, \ldots, u_{N_s})$ fast

Main question

Given a sequence $(u_n)_{n\geq 0}$ *in a ring* R, *and* $N \in \mathbb{N}$, *compute* u_N *fast*

▷ Input $(u_n)_{n\geq 0}$ is assumed to be a recurrent sequence, and it is specified by a recurrence relation and enough initial terms

▷ Efficiency is measured in terms of ring operations, or of bit operations

---

Two variants:

Given $(u_n)_n$ *in* $R^{\mathbb{N}}$ *and* $(N_1, \ldots, N_s) \in \mathbb{N}^s$, *compute* $(u_{N_1}, \ldots, u_{N_s})$ *fast*

and

Given $(u_n)_n$ *in* $\mathbb{Z}^{\mathbb{N}}$ *and* $(N_\ell)_{\ell=1}^s \in \mathbb{N}^s$, *compute* $(u_{N_\ell} \bmod N_\ell)_{\ell=1}^s$ *fast*

Examples

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

Examples

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

- geometric: $u_n = q^n$,

Examples

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

- geometric: $u_n = q^n$, i.e., $u_{n+1} = q \cdot u_n$ with $u_0 = 1$

Examples

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

- geometric: $u_n = q^n$, i.e., $u_{n+1} = q \cdot u_n$ with $u_0 = 1$
- Fibonacci: $u_{n+2} = u_{n+1} + u_n$ with $u_0 = u_1 = 1$

Examples

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

- geometric: $u_n = q^n$, i.e., $u_{n+1} = q \cdot u_n$ with $u_0 = 1$
- Fibonacci: $u_{n+2} = u_{n+1} + u_n$ with $u_0 = u_1 = 1$

C-recursive

Examples

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

- geometric: $u_n = q^n$, i.e., $u_{n+1} = q \cdot u_n$ with $u_0 = 1$
- Fibonacci: $u_{n+2} = u_{n+1} + u_n$ with $u_0 = u_1 = 1$

C-recursive

- factorial: $u_n = n!$,

Examples

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

- geometric: $u_n = q^n$, i.e., $u_{n+1} = q \cdot u_n$ with $u_0 = 1$
- Fibonacci: $u_{n+2} = u_{n+1} + u_n$ with $u_0 = u_1 = 1$

C-recursive

- factorial: $u_n = n!$, i.e., $u_{n+1} = (n+1) \cdot u_n$ with $u_0 = 1$

Examples

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

- geometric: $u_n = q^n$, i.e., $u_{n+1} = q \cdot u_n$ with $u_0 = 1$
- Fibonacci: $u_{n+2} = u_{n+1} + u_n$ with $u_0 = u_1 = 1$

C-recursive

- factorial: $u_n = n!$, i.e., $u_{n+1} = (n+1) \cdot u_n$ with $u_0 = 1$
- Motzkin: $u_{n+1} = \frac{2n+3}{n+3} \cdot u_n + \frac{3n}{n+3} \cdot u_{n-1}$ with $u_0 = u_1 = 1$

Examples

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

- geometric: $u_n = q^n$, i.e., $u_{n+1} = q \cdot u_n$ with $u_0 = 1$
- Fibonacci: $u_{n+2} = u_{n+1} + u_n$ with $u_0 = u_1 = 1$

C-recursive

- factorial: $u_n = n!$, i.e., $u_{n+1} = (n+1) \cdot u_n$ with $u_0 = 1$
- Motzkin: $u_{n+1} = \frac{2n+3}{n+3} \cdot u_n + \frac{3n}{n+3} \cdot u_{n-1}$ with $u_0 = u_1 = 1$

P-recursive (holonomic)

Examples

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

- geometric: $u_n = q^n$, i.e., $u_{n+1} = q \cdot u_n$ with $u_0 = 1$
- Fibonacci: $u_{n+2} = u_{n+1} + u_n$ with $u_0 = u_1 = 1$

C-recursive

- factorial: $u_n = n!$, i.e., $u_{n+1} = (n+1) \cdot u_n$ with $u_0 = 1$
- Motzkin: $u_{n+1} = \frac{2n+3}{n+3} \cdot u_n + \frac{3n}{n+3} \cdot u_{n-1}$ with $u_0 = u_1 = 1$

P-recursive (holonomic)

- q-factorial: $u_n = [n]_q! := (1+q)\cdots(1+q+\cdots+q^{n-1})$,

Examples

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

- geometric: $u_n = q^n$, i.e., $u_{n+1} = q \cdot u_n$ with $u_0 = 1$
- Fibonacci: $u_{n+2} = u_{n+1} + u_n$ with $u_0 = u_1 = 1$

C-recursive

- factorial: $u_n = n!$, i.e., $u_{n+1} = (n+1) \cdot u_n$ with $u_0 = 1$
- Motzkin: $u_{n+1} = \frac{2n+3}{n+3} \cdot u_n + \frac{3n}{n+3} \cdot u_{n-1}$ with $u_0 = u_1 = 1$

P-recursive (holonomic)

- q-factorial: $u_n = [n]_q! := (1+q)\cdots(1+q+\cdots+q^{n-1})$,
  i.e., $u_{n+1} = (1+q+\cdots+q^n) \cdot u_n$ with $u_0 = 1$

Examples

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

- geometric: $u_n = q^n$, i.e., $u_{n+1} = q \cdot u_n$ with $u_0 = 1$
- Fibonacci: $u_{n+2} = u_{n+1} + u_n$ with $u_0 = u_1 = 1$

C-recursive

- factorial: $u_n = n!$, i.e., $u_{n+1} = (n+1) \cdot u_n$ with $u_0 = 1$
- Motzkin: $u_{n+1} = \frac{2n+3}{n+3} \cdot u_n + \frac{3n}{n+3} \cdot u_{n-1}$ with $u_0 = u_1 = 1$

P-recursive (holonomic)

- q-factorial: $u_n = [n]_q! := (1+q)\cdots(1+q+\cdots+q^{n-1})$,
  i.e., $u_{n+1} = (1+q+\cdots+q^n) \cdot u_n$ with $u_0 = 1$
- $\sum_{k=0}^{n-1} q^{k^2}$: $u_{n+1} - u_n = q^{2n-1}(u_n - u_{n-1})$ with $u_0=0, u_1=1$

Examples

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

- geometric: $u_n = q^n$, i.e., $u_{n+1} = q \cdot u_n$ with $u_0 = 1$
- Fibonacci: $u_{n+2} = u_{n+1} + u_n$ with $u_0 = u_1 = 1$

C-recursive

- factorial: $u_n = n!$, i.e., $u_{n+1} = (n+1) \cdot u_n$ with $u_0 = 1$
- Motzkin: $u_{n+1} = \frac{2n+3}{n+3} \cdot u_n + \frac{3n}{n+3} \cdot u_{n-1}$ with $u_0 = u_1 = 1$

P-recursive (holonomic)

- q-factorial: $u_n = [n]_q! := (1+q)\cdots(1+q+\cdots+q^{n-1})$,
  i.e., $u_{n+1} = (1+q+\cdots+q^n) \cdot u_n$ with $u_0 = 1$
- $\sum_{k=0}^{n-1} q^{k^2}$: $u_{n+1} - u_n = q^{2n-1}(u_n - u_{n-1})$ with $u_0 = 0, u_1 = 1$

q-holonomic

Examples

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

- geometric: $u_n = q^n$, i.e., $u_{n+1} = q \cdot u_n$ with $u_0 = 1$
- Fibonacci: $u_{n+2} = u_{n+1} + u_n$ with $u_0 = u_1 = 1$

C-recursive

- factorial: $u_n = n!$, i.e., $u_{n+1} = (n+1) \cdot u_n$ with $u_0 = 1$
- Motzkin: $u_{n+1} = \frac{2n+3}{n+3} \cdot u_n + \frac{3n}{n+3} \cdot u_{n-1}$ with $u_0 = u_1 = 1$

P-recursive (holonomic)

- q-factorial: $u_n = [n]_q! := (1+q) \cdots (1+q+\cdots+q^{n-1})$,
  i.e., $u_{n+1} = (1+q+\cdots+q^n) \cdot u_n$ with $u_0 = 1$
- $\sum_{k=0}^{n-1} q^{k^2}$: $u_{n+1} - u_n = q^{2n-1}(u_n - u_{n-1})$ with $u_0 = 0, u_1 = 1$

q-holonomic

- Göbel: $u_{n+1} = \frac{1}{n} \cdot (1 + u_0^2 + u_1^2 + \cdots + u_{n-1}^2)$ with $u_0 = 1$

Examples

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

- geometric: $u_n = q^n$, i.e., $u_{n+1} = q \cdot u_n$ with $u_0 = 1$
- Fibonacci: $u_{n+2} = u_{n+1} + u_n$ with $u_0 = u_1 = 1$

C-recursive

- factorial: $u_n = n!$, i.e., $u_{n+1} = (n+1) \cdot u_n$ with $u_0 = 1$
- Motzkin: $u_{n+1} = \frac{2n+3}{n+3} \cdot u_n + \frac{3n}{n+3} \cdot u_{n-1}$ with $u_0 = u_1 = 1$

P-recursive (holonomic)

- q-factorial: $u_n = [n]_q! := (1+q)\cdots(1+q+\cdots+q^{n-1})$,
  i.e., $u_{n+1} = (1+q+\cdots+q^n) \cdot u_n$ with $u_0 = 1$
- $\sum_{k=0}^{n-1} q^{k^2}$: $u_{n+1} - u_n = q^{2n-1}(u_n - u_{n-1})$ with $u_0 = 0, u_1 = 1$

q-holonomic

- Göbel: $u_{n+1} = \frac{1}{n} \cdot (1 + u_0^2 + u_1^2 + \cdots + u_{n-1}^2)$ with $u_0 = 1$
- Somos: $u_{n+5} = \frac{u_{n+4} \cdot u_{n+1} + u_{n+3} \cdot u_{n+2}}{u_n}$ with $u_0 = \cdots = u_4 = 1$

Examples

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

- geometric: $u_n = q^n$, i.e., $u_{n+1} = q \cdot u_n$ with $u_0 = 1$
- Fibonacci: $u_{n+2} = u_{n+1} + u_n$ with $u_0 = u_1 = 1$

C-recursive

- factorial: $u_n = n!$, i.e., $u_{n+1} = (n+1) \cdot u_n$ with $u_0 = 1$
- Motzkin: $u_{n+1} = \frac{2n+3}{n+3} \cdot u_n + \frac{3n}{n+3} \cdot u_{n-1}$ with $u_0 = u_1 = 1$

P-recursive (holonomic)

- q-factorial: $u_n = [n]_q! := (1+q)\cdots(1+q+\cdots+q^{n-1})$,
  i.e., $u_{n+1} = (1+q+\cdots+q^n) \cdot u_n$ with $u_0 = 1$
- $\sum_{k=0}^{n-1} q^{k^2}$: $u_{n+1} - u_n = q^{2n-1}(u_n - u_{n-1})$ with $u_0=0, u_1=1$

q-holonomic

- Göbel: $u_{n+1} = \frac{1}{n} \cdot (1 + u_0^2 + u_1^2 + \cdots + u_{n-1}^2)$ with $u_0 = 1$
- Somos: $u_{n+5} = \frac{u_{n+4} \cdot u_{n+1} + u_{n+3} \cdot u_{n+2}}{u_n}$ with $u_0 = \cdots = u_4 = 1$

non-linear

Examples

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

- geometric: $u_n = q^n$, i.e., $u_{n+1} = q \cdot u_n$ with $u_0 = 1$
- Fibonacci: $u_{n+2} = u_{n+1} + u_n$ with $u_0 = u_1 = 1$

C-recursive

- factorial: $u_n = n!$, i.e., $u_{n+1} = (n+1) \cdot u_n$ with $u_0 = 1$
- Motzkin: $u_{n+1} = \frac{2n+3}{n+3} \cdot u_n + \frac{3n}{n+3} \cdot u_{n-1}$ with $u_0 = u_1 = 1$

P-recursive (holonomic)

- q-factorial: $u_n = [n]_q! := (1+q)\cdots(1+q+\cdots+q^{n-1})$,
  i.e., $u_{n+1} = (1+q+\cdots+q^n) \cdot u_n$ with $u_0 = 1$
- $\sum_{k=0}^{n-1} q^{k^2}$: $u_{n+1} - u_n = q^{2n-1}(u_n - u_{n-1})$ with $u_0 = 0, u_1 = 1$

q-holonomic

- Göbel: $u_{n+1} = \frac{1}{n} \cdot (1 + u_0^2 + u_1^2 + \cdots + u_{n-1}^2)$ with $u_0 = 1$
- Somos: $u_{n+5} = \frac{u_{n+4} \cdot u_{n+1} + u_{n+3} \cdot u_{n+2}}{u_n}$ with $u_0 = \cdots = u_4 = 1$

non-linear

- Katz: $u_{n+1} = \frac{\partial u_n}{\partial x} - M \cdot u_n$ with $M \in \mathcal{M}_r(\mathbb{F}_p(x))$, $u_0 = I_r$

Examples

Given a sequence $(u_n)_{n\geq 0}$ in a ring R, and $N \in \mathbb{N}$, compute u_N fast

- geometric: $u_n = q^n$, i.e., $u_{n+1} = q \cdot u_n$ with $u_0 = 1$
- Fibonacci: $u_{n+2} = u_{n+1} + u_n$ with $u_0 = u_1 = 1$

C-recursive

- factorial: $u_n = n!$, i.e., $u_{n+1} = (n+1) \cdot u_n$ with $u_0 = 1$
- Motzkin: $u_{n+1} = \frac{2n+3}{n+3} \cdot u_n + \frac{3n}{n+3} \cdot u_{n-1}$ with $u_0 = u_1 = 1$

P-recursive (holonomic)

- q-factorial: $u_n = [n]_q! := (1+q)\cdots(1+q+\cdots+q^{n-1})$,
  i.e., $u_{n+1} = (1+q+\cdots+q^n) \cdot u_n$ with $u_0 = 1$
- $\sum_{k=0}^{n-1} q^{k^2}$: $u_{n+1} - u_n = q^{2n-1}(u_n - u_{n-1})$ with $u_0 = 0, u_1 = 1$

q-holonomic

- Göbel: $u_{n+1} = \frac{1}{n} \cdot (1 + u_0^2 + u_1^2 + \cdots + u_{n-1}^2)$ with $u_0 = 1$
- Somos: $u_{n+5} = \frac{u_{n+4} \cdot u_{n+1} + u_{n+3} \cdot u_{n+2}}{u_n}$ with $u_0 = \cdots = u_4 = 1$

non-linear

- Katz: $u_{n+1} = \frac{\partial u_n}{\partial x} - M \cdot u_n$ with $M \in \mathcal{M}_r(\mathbb{F}_p(x))$, $u_0 = I_r$

p-curvature

Motivations

- algebraic complexity theory

Motivations

- algebraic complexity theory
  - *evaluation of polynomials*: x^N and $\sum_{\ell=0}^{N} 2^\ell x^\ell$ vs. $\sum_{\ell=0}^{N} 2^{2^\ell} x^\ell$ [Strassen, 1974]

Motivations

- algebraic complexity theory
  - *evaluation of polynomials*: x^N and $\sum_{\ell=0}^{N} 2^\ell x^\ell$ vs. $\sum_{\ell=0}^{N} 2^{2^\ell} x^\ell$ [Strassen, 1974]
- basic computer algebra questions

Motivations

- algebraic complexity theory
  - *evaluation of polynomials*: x^N and $\sum_{\ell=0}^{N} 2^\ell x^\ell$ vs. $\sum_{\ell=0}^{N} 2^{2^\ell} x^\ell$ [Strassen, 1974]
- basic computer algebra questions
  - *matrix powering* M^N; more generally, $P(M)$ [Giesbrecht, 1995]

Motivations

- algebraic complexity theory
  - *evaluation of polynomials*: x^N and $\sum_{\ell=0}^{N} 2^\ell x^\ell$ vs. $\sum_{\ell=0}^{N} 2^{2^\ell} x^\ell$ [Strassen, 1974]
- basic computer algebra questions
  - *matrix powering* M^N; more generally, $P(M)$ [Giesbrecht, 1995]
  - *Graeffe polynomials* $\prod_{P(\alpha)=0} (x - \alpha^N)$ [B., Flajolet, Salvy, Schost, 2006]

Motivations

- algebraic complexity theory
  - *evaluation of polynomials*: x^N and $\sum_{\ell=0}^{N} 2^\ell x^\ell$ vs. $\sum_{\ell=0}^{N} 2^{2^\ell} x^\ell$ [Strassen, 1974]
- basic computer algebra questions
  - *matrix powering* M^N; more generally, $P(M)$ [Giesbrecht, 1995]
  - *Graeffe polynomials* $\prod_{P(\alpha)=0} (x - \alpha^N)$ [B., Flajolet, Salvy, Schost, 2006]
  - *modular polynomial exponentiation* $P^N \bmod Q$ [B., Mori, 2020]

Motivations

- algebraic complexity theory
  - *evaluation of polynomials*: x^N and $\sum_{\ell=0}^{N} 2^\ell x^\ell$ vs. $\sum_{\ell=0}^{N} 2^{2^\ell} x^\ell$ [Strassen, 1974]
- basic computer algebra questions
  - *matrix powering* M^N; more generally, $P(M)$ [Giesbrecht, 1995]
  - *Graeffe polynomials* $\prod_{P(\alpha)=0} (x - \alpha^N)$ [B., Flajolet, Salvy, Schost, 2006]
  - *modular polynomial exponentiation* $P^N \bmod Q$ [B., Mori, 2020]
- more involved computer algebra questions

Motivations

- algebraic complexity theory
  - *evaluation of polynomials*: x^N and $\sum_{\ell=0}^{N} 2^\ell x^\ell$ vs. $\sum_{\ell=0}^{N} 2^{2^\ell} x^\ell$ [Strassen, 1974]
- basic computer algebra questions
  - *matrix powering* M^N; more generally, $P(M)$ [Giesbrecht, 1995]
  - *Graeffe polynomials* $\prod_{P(\alpha)=0} (x - \alpha^N)$ [B., Flajolet, Salvy, Schost, 2006]
  - *modular polynomial exponentiation* $P^N \bmod Q$ [B., Mori, 2020]
- more involved computer algebra questions
  - *polynomial linear algebra* [Storjohann, 2003]

Motivations

- algebraic complexity theory
  - *evaluation of polynomials*: x^N and $\sum_{\ell=0}^{N} 2^\ell x^\ell$ vs. $\sum_{\ell=0}^{N} 2^{2^\ell} x^\ell$ [Strassen, 1974]
- basic computer algebra questions
  - *matrix powering* M^N; more generally, $P(M)$ [Giesbrecht, 1995]
  - *Graeffe polynomials* $\prod_{P(\alpha)=0} (x - \alpha^N)$ [B., Flajolet, Salvy, Schost, 2006]
  - *modular polynomial exponentiation* $P^N \bmod Q$ [B., Mori, 2020]
- more involved computer algebra questions
  - *polynomial linear algebra* [Storjohann, 2003]
  - *factoring in* $\mathbb{F}_q[x]$ [Berlekamp, 1970; Cantor, Zassenhaus, 1981; Shoup, 1995]

Motivations

- algebraic complexity theory
  - *evaluation of polynomials*: x^N and $\sum_{\ell=0}^{N} 2^\ell x^\ell$ vs. $\sum_{\ell=0}^{N} 2^{2^\ell} x^\ell$ [Strassen, 1974]

- basic computer algebra questions
  - *matrix powering* M^N; more generally, $P(M)$ [Giesbrecht, 1995]
  - *Graeffe polynomials* $\prod_{P(\alpha)=0} (x - \alpha^N)$ [B., Flajolet, Salvy, Schost, 2006]
  - *modular polynomial exponentiation* $P^N \bmod Q$ [B., Mori, 2020]

- more involved computer algebra questions
  - *polynomial linear algebra* [Storjohann, 2003]
  - *factoring in* $\mathbb{F}_q[x]$ [Berlekamp, 1970; Cantor, Zassenhaus, 1981; Shoup, 1995]

- algorithmic number theory

Motivations

- algebraic complexity theory
  - *evaluation of polynomials*: x^N and $\sum_{\ell=0}^{N} 2^\ell x^\ell$ vs. $\sum_{\ell=0}^{N} 2^{2^\ell} x^\ell$ [Strassen, 1974]

- basic computer algebra questions
  - *matrix powering* M^N; more generally, $P(M)$ [Giesbrecht, 1995]
  - *Graeffe polynomials* $\prod_{P(\alpha)=0} (x - \alpha^N)$ [B., Flajolet, Salvy, Schost, 2006]
  - *modular polynomial exponentiation* $P^N \bmod Q$ [B., Mori, 2020]

- more involved computer algebra questions
  - *polynomial linear algebra* [Storjohann, 2003]
  - *factoring in* $\mathbb{F}_q[x]$ [Berlekamp, 1970; Cantor, Zassenhaus, 1981; Shoup, 1995]

- algorithmic number theory
  - *primality tests* [Solovay, Strassen, 1977; Miller, 1976; Rabin, 1980; Atkin, Morain, 1994; Agrawal, Kayal, Saxena, 2004]

Motivations

- algebraic complexity theory
  - *evaluation of polynomials*: x^N and $\sum_{\ell=0}^N 2^\ell x^\ell$ vs. $\sum_{\ell=0}^N 2^{2^\ell} x^\ell$ [Strassen, 1974]
- basic computer algebra questions
  - *matrix powering* M^N; more generally, $P(M)$ [Giesbrecht, 1995]
  - *Graeffe polynomials* $\prod_{P(\alpha)=0} (x - \alpha^N)$ [B., Flajolet, Salvy, Schost, 2006]
  - *modular polynomial exponentiation* $P^N \bmod Q$ [B., Mori, 2020]
- more involved computer algebra questions
  - *polynomial linear algebra* [Storjohann, 2003]
  - *factoring in* $\mathbb{F}_q[x]$ [Berlekamp, 1970; Cantor, Zassenhaus, 1981; Shoup, 1995]
- algorithmic number theory
  - *primality tests* [Solovay, Strassen, 1977; Miller, 1976; Rabin, 1980; Atkin, Morain, 1994; Agrawal, Kayal, Saxena, 2004]
- effective algebraic geometry

Motivations

- algebraic complexity theory
  - *evaluation of polynomials*: x^N and $\sum_{\ell=0}^{N} 2^\ell x^\ell$ vs. $\sum_{\ell=0}^{N} 2^{2^\ell} x^\ell$ [Strassen, 1974]
- basic computer algebra questions
  - *matrix powering* M^N; more generally, $P(M)$ [Giesbrecht, 1995]
  - *Graeffe polynomials* $\prod_{P(\alpha)=0} (x - \alpha^N)$ [B., Flajolet, Salvy, Schost, 2006]
  - *modular polynomial exponentiation* $P^N \bmod Q$ [B., Mori, 2020]
- more involved computer algebra questions
  - *polynomial linear algebra* [Storjohann, 2003]
  - *factoring in* $\mathbb{F}_q[x]$ [Berlekamp, 1970; Cantor, Zassenhaus, 1981; Shoup, 1995]
- algorithmic number theory
  - *primality tests* [Solovay, Strassen, 1977; Miller, 1976; Rabin, 1980; Atkin, Morain, 1994; Agrawal, Kayal, Saxena, 2004]
- effective algebraic geometry
  - *counting points on elliptic curves* over $\mathbb{F}_q$ [Schoof-Elkies-Atkin, 1992–1998]

Overview: naive algorithms

Seq. Term	Arith. size	Arith. cost	Method	Bit size	Bit cost	Method
q^N	1	$O(N)$	iterative algorithm	N	$\tilde{O}(N^2)$	iterative algorithm

† assuming quasi-optimal (FFT-based) integer multiplication $\mathsf{M}(N) = \tilde{O}(N)$

Overview: naive algorithms

Seq. Term	Arith. size	Arith. cost	Method	Bit size	Bit cost	Method
q^N F_N	1 1	$O(N)$ $O(N)$	iterative algorithm	N N	$\tilde{O}(N^2)$ $\tilde{O}(N^2)$	iterative algorithm

† assuming quasi-optimal (FFT-based) integer multiplication $\mathsf{M}(N) = \tilde{O}(N)$

Overview: naive algorithms

Seq. Term	Arith. size	Arith. cost	Method	Bit size	Bit cost	Method
q^N	1	$O(N)$	iterative algorithm	N	$\tilde{O}(N^2)$	iterative algorithm
F_N	1	$O(N)$		N	$\tilde{O}(N^2)$	
$N!$	1	$O(N)$	iterative algorithm	$N \log N$	$\tilde{O}(N^2)$	iterative algorithm

† assuming quasi-optimal (FFT-based) integer multiplication $\mathsf{M}(N) = \tilde{O}(N)$

Overview: naive algorithms

Seq. Term	Arith. size	Arith. cost	Method	Bit size	Bit cost	Method
q^N	1	$O(N)$	iterative	N	$\tilde{O}(N^2)$	iterative
F_N	1	$O(N)$	algorithm	N	$\tilde{O}(N^2)$	algorithm
$N!$	1	$O(N)$	iterative	$N \log N$	$\tilde{O}(N^2)$	iterative
M_N	1	$O(N)$	algorithm	$N \log N$	$\tilde{O}(N^2)$	algorithm

† assuming quasi-optimal (FFT-based) integer multiplication $\mathsf{M}(N) = \tilde{O}(N)$

Overview: naive algorithms

Seq. Term	Arith. size	Arith. cost	Method	Bit size	Bit cost	Method
q^N	1	$O(N)$	iterative	N	$\tilde{O}(N^2)$	iterative
F_N	1	$O(N)$	algorithm	N	$\tilde{O}(N^2)$	algorithm
$N!$	1	$O(N)$	iterative	$N \log N$	$\tilde{O}(N^2)$	iterative
M_N	1	$O(N)$	algorithm	$N \log N$	$\tilde{O}(N^2)$	algorithm
$[N]_q!$	1	$O(N)$	iterative algorithm	N^2	$\tilde{O}(N^3)$	iterative algorithm

† assuming quasi-optimal (FFT-based) integer multiplication $\mathsf{M}(N) = \tilde{O}(N)$

Overview: naive algorithms

Seq. Term	Arith. size	Arith. cost	Method	Bit size	Bit cost	Method
q^N	1	$O(N)$	iterative	N	$\tilde{O}(N^2)$	iterative
F_N	1	$O(N)$	algorithm	N	$\tilde{O}(N^2)$	algorithm
$N!$	1	$O(N)$	iterative	$N \log N$	$\tilde{O}(N^2)$	iterative
M_N	1	$O(N)$	algorithm	$N \log N$	$\tilde{O}(N^2)$	algorithm
$[N]_q!$	1	$O(N)$	iterative	N^2	$\tilde{O}(N^3)$	iterative
$\sum_{n=0}^{N} q^{n^2}$	1	$O(N^2)$	algorithm	N^2	$\tilde{O}(N^3)$	algorithm

† assuming quasi-optimal (FFT-based) integer multiplication $\mathsf{M}(N) = \tilde{O}(N)$

Overview: best algorithms

Seq. Term	Arith. size	Arith. cost	Method	Bit size	Bit cost	Method
q^N	1	$O(\log N)$	binary powering	N	$\tilde{O}(N)$	binary powering

† assuming FFT-based integer and polynomial multiplication $\mathsf{M}(N) = \tilde{O}(N)$

Seq. Term	Arith. size	Arith. cost	Method	Bit size	Bit cost	Method
q^N F_N	1 1	$O(\log N)$ $O(\log N)$	binary powering	N N	$\tilde{O}(N)$ $\tilde{O}(N)$	binary powering

† assuming FFT-based integer and polynomial multiplication $\mathsf{M}(N) = \tilde{O}(N)$

Overview: best algorithms

Seq. Term	Arith. size	Arith. cost	Method	Bit size	Bit cost	Method
q^N F_N	1 1	$O(\log N)$ $O(\log N)$	binary powering	N N	$\tilde{O}(N)$ $\tilde{O}(N)$	binary powering
$N!$	1	$\tilde{O}(\sqrt{N})$	baby-steps / giant-steps	$N \log N$	$\tilde{O}(N)$	binary splitting

† assuming FFT-based integer and polynomial multiplication $\mathsf{M}(N) = \tilde{O}(N)$

Overview: best algorithms

Seq. Term	Arith. size	Arith. cost	Method	Bit size	Bit cost	Method
q^N	1	$O(\log N)$	binary	N	$\tilde{O}(N)$	binary
F_N	1	$O(\log N)$	powering	N	$\tilde{O}(N)$	powering
$N!$	1	$\tilde{O}(\sqrt{N})$	baby-steps /	$N \log N$	$\tilde{O}(N)$	binary
M_N	1	$\tilde{O}(\sqrt{N})$	giant-steps	$N \log N$	$\tilde{O}(N)$	splitting

† assuming FFT-based integer and polynomial multiplication $\mathsf{M}(N) = \tilde{O}(N)$

Overview: best algorithms

Seq. Term	Arith. size	Arith. cost	Method	Bit size	Bit cost	Method
q^N	1	$O(\log N)$	binary	N	$\tilde{O}(N)$	binary
F_N	1	$O(\log N)$	powering	N	$\tilde{O}(N)$	powering
$N!$	1	$\tilde{O}(\sqrt{N})$	baby-steps /	$N \log N$	$\tilde{O}(N)$	binary
M_N	1	$\tilde{O}(\sqrt{N})$	giant-steps	$N \log N$	$\tilde{O}(N)$	splitting
$[N]_q!$	1	$\tilde{O}(\sqrt{N})$	baby-steps / giant-steps	N^2	$\tilde{O}(N^2)$	binary splitting

† assuming FFT-based integer and polynomial multiplication $\mathsf{M}(N) = \tilde{O}(N)$

Overview: best algorithms

Seq. Term	Arith. size	Arith. cost	Method	Bit size	Bit cost	Method
q^N	1	$O(\log N)$	binary	N	$\tilde{O}(N)$	binary
F_N	1	$O(\log N)$	powering	N	$\tilde{O}(N)$	powering
$N!$	1	$\tilde{O}(\sqrt{N})$	baby-steps /	$N \log N$	$\tilde{O}(N)$	binary
M_N	1	$\tilde{O}(\sqrt{N})$	giant-steps	$N \log N$	$\tilde{O}(N)$	splitting
$[N]_q!$	1	$\tilde{O}(\sqrt{N})$	baby-steps /	N^2	$\tilde{O}(N^2)$	binary
$\sum_{n=0}^{N-1} q^{n^2}$	1	$\tilde{O}(\sqrt{N})$	giant-steps	N^2	$\tilde{O}(N^2)$	splitting

† assuming FFT-based integer and polynomial multiplication $\mathsf{M}(N) = \tilde{O}(N)$

Overview: best algorithms

Seq. Term	Arith. size	Arith. cost	Method	Bit size	Bit cost	Method
q^N	1	$O(\log N)$	binary	N	$\tilde{O}(N)$	binary
F_N	1	$O(\log N)$	powering	N	$\tilde{O}(N)$	powering
$N!$	1	$\tilde{O}(\sqrt{N})$	baby-steps /	$N \log N$	$\tilde{O}(N)$	binary
M_N	1	$\tilde{O}(\sqrt{N})$	giant-steps	$N \log N$	$\tilde{O}(N)$	splitting
$[N]_q!$	1	$\tilde{O}(\sqrt{N})$	baby-steps /	N^2	$\tilde{O}(N^2)$	binary
$\sum_{n=0}^{N-1} q^{n^2}$	1	$\tilde{O}(\sqrt{N})$	giant-steps	N^2	$\tilde{O}(N^2)$	splitting

▷ For $R = \mathbb{F}_p$: $M_N \in R$ in $O(\log N)$ ops. in R; same for any u_N with *algebraic* GF $\sum_n u_n x^n$ [B., Christol, Dumas, 2016], [B., Caruso, Christol, Dumas, 2019]

† assuming FFT-based integer and polynomial multiplication $\mathsf{M}(N) = \tilde{O}(N)$

Overview: best algorithms

Seq. Term	Arith. size	Arith. cost	Method	Bit size	Bit cost	Method
q^N	1	$O(\log N)$	binary	N	$\tilde{O}(N)$	binary
F_N	1	$O(\log N)$	powering	N	$\tilde{O}(N)$	powering
$N!$	1	$\tilde{O}(\sqrt{N})$	baby-steps /	$N \log N$	$\tilde{O}(N)$	binary
M_N	1	$\tilde{O}(\sqrt{N})$	giant-steps	$N \log N$	$\tilde{O}(N)$	splitting
$[N]_q!$	1	$\tilde{O}(\sqrt{N})$	baby-steps /	N^2	$\tilde{O}(N^2)$	binary
$\sum_{n=0}^{N-1} q^{n^2}$	1	$\tilde{O}(\sqrt{N})$	giant-steps	N^2	$\tilde{O}(N^2)$	splitting

- ▷ First part of this course focusses on the first two rows
- ▷ Second part: two middle rows
- ▷ Bonus: last two rows

† assuming FFT-based integer and polynomial multiplication $\mathsf{M}(N) = \tilde{O}(N)$

COMPUTING TERMS OF C-RECURSIVE SEQUENCES

Main question

Given a sequence $(u_n)_{n\geq 0}$ in a field $\mathbb{K}$, and $N \in \mathbb{N}$, compute u_N fast

Main question

Given a sequence $(u_n)_{n\geq 0}$ *in a field* $\mathbb{K}$*, and* $N \in \mathbb{N}$*, compute* u_N *fast*

▷ Input $(u_n)_{n\geq 0}$ is assumed to be a recurrent sequence, and it is specified by a recurrence relation and enough initial terms

Main question

Given a sequence $(u_n)_{n\geq 0}$ *in a field* $\mathbb{K}$*, and* $N \in \mathbb{N}$*, compute* u_N *fast*

▷ Input $(u_n)_{n\geq 0}$ is assumed to be a recurrent sequence, and it is specified by a recurrence relation and enough initial terms

▷ Efficiency is measured in terms of field operations (arithmetic complexity)

Main question

Given a sequence $(u_n)_{n\geq 0}$ in a field $\mathbb{K}$, and $N \in \mathbb{N}$, compute u_N fast

▷ Input $(u_n)_{n\geq 0}$ is assumed to be a recurrent sequence, and it is specified by a recurrence relation and enough initial terms

▷ Efficiency is measured in terms of field operations (arithmetic complexity)

First part: input sequence is C-recursive, given by initial terms $u_0, \dots, u_{d-1}$ and a linear recurrence with constant coefficients $(c_0, \dots, c_{d-1}) \in \mathbb{K}^d$

$$u_{n+d} = c_{d-1}u_{n+d-1} + \cdots + c_0 u_n, \qquad n \geq 0.$$

Main question

Given a sequence $(u_n)_{n\geq 0}$ in a field $\mathbb{K}$, and $N \in \mathbb{N}$, compute u_N fast

▷ Input $(u_n)_{n\geq 0}$ is assumed to be a recurrent sequence, and it is specified by a recurrence relation and enough initial terms

▷ Efficiency is measured in terms of field operations (arithmetic complexity)

First part: input sequence is C-recursive, given by initial terms $u_0, \dots, u_{d-1}$ and a linear recurrence with constant coefficients $(c_0, \dots, c_{d-1}) \in \mathbb{K}^d$

$$u_{n+d} = c_{d-1}u_{n+d-1} + \cdots + c_0 u_n, \qquad n \geq 0.$$

▷ Def. $\Gamma(x) := x^d - \sum_{i=0}^{d-1} c_i x^i$ is called *characteristic polynomial* for $(u_n)_{n\geq 0}$

Binary powering

Problem: Given a ring R, $a \in R$ and $N \geq 1$, compute a^N

Binary powering

Problem: Given a ring R, $a \in R$ and $N \geq 1$, compute a^N

▷ Naive (iterative) algorithm: $O(N)$ ops. in R

Problem: Given a ring R, $a \in R$ and $N \geq 1$, compute a^N

▷ Naive (iterative) algorithm: $O(N)$ ops. in R

▷ Better algorithm (Pingala, 200 BC): $O(\log N)$ ops. in R

Compute a^N recursively, using square-and-multiply

$$a^N = \begin{cases} (a^{N/2})^2, & \text{if } N \text{ is even,} \\ a \cdot (a^{\frac{N-1}{2}})^2, & \text{else.} \end{cases}$$

Binary powering

Problem: Given a ring R, $a \in R$ and $N \geq 1$, compute a^N

▷ Naive (iterative) algorithm: $O(N)$ ops. in R

▷ Better algorithm (Pingala, 200 BC): $O(\log N)$ ops. in R

Compute a^N recursively, using square-and-multiply

$$a^N = \begin{cases} (a^{N/2})^2, & \text{if } N \text{ is even,} \\ a \cdot (a^{\frac{N-1}{2}})^2, & \text{else.} \end{cases}$$

▷ Application: modular exponentiation in $R = \mathcal{M}_d(\mathbb{K})$:

- if $M \in \mathcal{M}_d(\mathbb{K})$, then M^N in $O(d^\theta \log N)$ ops. in $\mathbb{K}$

Problem: Given a ring R, $a \in R$ and $N \geq 1$, compute a^N

▷ Naive (iterative) algorithm: $O(N)$ ops. in R

▷ Better algorithm (Pingala, 200 BC): $O(\log N)$ ops. in R

Compute a^N recursively, using square-and-multiply

$$a^N = \begin{cases} (a^{N/2})^2, & \text{if } N \text{ is even,} \\ a \cdot (a^{\frac{N-1}{2}})^2, & \text{else.} \end{cases}$$

▷ Application: modular exponentiation in $R = \mathbb{K}[x]/(Q)$:

- if $P, Q \in \mathbb{K}[x]_{<d}$, then $P^N \bmod Q$ in $O(\mathsf{M}(d) \log N)$ ops. in $\mathbb{K}$

Binary powering

Problem: Given a ring R, $a \in R$ and $N \geq 1$, compute a^N

▷ Naive (iterative) algorithm: $O(N)$ ops. in R

▷ Better algorithm (Pingala, 200 BC): $O(\log N)$ ops. in R

Compute a^N recursively, using square-and-multiply

$$a^N = \begin{cases} (a^{N/2})^2, & \text{if } N \text{ is even,} \\ a \cdot (a^{\frac{N-1}{2}})^2, & \text{else.} \end{cases}$$

▷ Application: modular exponentiation in $R = \mathbb{Z}/A\mathbb{Z}$:

- N-th decimal of $\frac{1}{A}$ via $(10^{N-1} \bmod A)$ in $O(\mathsf{M}_{\mathbb{Z}}(\log A) \log N)$ bit ops.

Example: N-th decimal of a rational number

What is the 10^{10^6}-th decimal of $A = \frac{1}{2039}$?

Example: N-th decimal of a rational number

What is the 10^{10^6}-th decimal of $A = \frac{1}{2039}$?

```
> N:=10^(10^6): A:=2039:
> iquo(10*(irem(10^(N-1),A)), A);
```

Example: N-th decimal of a rational number

What is the 10^{10^6}-th decimal of $A = \frac{1}{2039}$?

```
> N:=10^(10^6): A:=2039:
> iquo(10*(irem(10^(N-1),A)), A);
```

Error, numeric exception: overflow

Example: N-th decimal of a rational number

What is the 10^{10^6}-th decimal of $A = \frac{1}{2039}$?

```
> N:=10^(10^6): A:=2039:
> iquo(10*(irem(10^(N-1),A)), A);
```

Error, numeric exception: overflow

```
> st:=time(): iquo(10*(`&^`(10,N-1) mod A), A), time()-st;
```

Example: N-th decimal of a rational number

What is the 10^{10^6}-th decimal of $A = \frac{1}{2039}$?

```
> N:=10^(10^6): A:=2039:
> iquo(10*(irem(10^(N-1),A)), A);
```

Error, numeric exception: overflow

```
> st:=time(): iquo(10*(`&^`(10,N-1) mod A), A), time()-st;
```

6, 0.037

Example: N-th decimal of a rational number

What is the 10^{10^6}-th decimal of $A = \frac{1}{2039}$?

```
> N:=10^(10^6): A:=2039:
> iquo(10*(irem(10^(N-1),A)), A);
```

Error, numeric exception: overflow

```
> st:=time(): iquo(10*(`&^`(10,N-1) mod A), A), time()-st;
```

6, 0.037

▷ The following also computes the right answer. Can you see why?

```
> n := irem(N,A-1);
> iquo(10*(irem(10^(n-1),A)), A);
```

6

RULE 1: *Do care about the size of $\mathcal{O}$!*

Fast polynomial division and modular exponentiation [Strassen, 1973]

Pb: Given $F \in \mathbb{K}[x]_{<2d}$ and $Q \in \mathbb{K}[x]_d$ compute (U, R) in Euclidean division

$$F = UQ + R$$

Fast polynomial division and modular exponentiation [Strassen, 1973]

Pb: Given $F \in \mathbb{K}[x]_{<2d}$ and $Q \in \mathbb{K}[x]_d$ compute (U, R) in Euclidean division

$$F = UQ + R$$

Naive algorithm: $O(d^2)$

Fast polynomial division and modular exponentiation [Strassen, 1973]

Pb: Given $F \in \mathbb{K}[x]_{<2d}$ and $Q \in \mathbb{K}[x]_d$ compute (U, R) in Euclidean division

$$F = UQ + R$$

Naive algorithm: $O(d^2)$

Idea: when $\mathbb{K} = \mathbb{R}$, look at $F = UQ + R$ from infinity: $U \sim_{+\infty} F/Q$

Pb: Given $F \in \mathbb{K}[x]_{<2d}$ and $Q \in \mathbb{K}[x]_d$ compute (U, R) in Euclidean division

$$F = UQ + R$$

Naive algorithm: $O(d^2)$

Idea: when $\mathbb{K} = \mathbb{R}$, look at $F = UQ + R$ from infinity: $U \sim_{+\infty} F/Q$

Formalization: Let $D = \deg(F)$. Then $\deg(U) = D - d < d$, $\deg(R) < d$ and

$$\underbrace{F(1/x)x^D}_{\mathrm{rev}(F)} = \underbrace{Q(1/x)x^d}_{\mathrm{rev}(Q)} \cdot \underbrace{U(1/x)x^{D-d}}_{\mathrm{rev}(U)} + \underbrace{R(1/x)x^{\deg(R)}}_{\mathrm{rev}(R)} \cdot x^{D-\deg(R)}$$

Fast polynomial division and modular exponentiation [Strassen, 1973]

Pb: Given $F \in \mathbb{K}[x]_{<2d}$ and $Q \in \mathbb{K}[x]_d$ compute (U, R) in Euclidean division

$$F = UQ + R$$

Naive algorithm: $O(d^2)$

Idea: when $\mathbb{K} = \mathbb{R}$, look at $F = UQ + R$ from infinity: $U \sim_{+\infty} F/Q$

Formalization: Let $D = \deg(F)$. Then $\deg(U) = D - d < d$, $\deg(R) < d$ and

$$\underbrace{F(1/x)x^D}_{\mathrm{rev}(F)} = \underbrace{Q(1/x)x^d}_{\mathrm{rev}(Q)} \cdot \underbrace{U(1/x)x^{D-d}}_{\mathrm{rev}(U)} + \underbrace{R(1/x)x^{\deg(R)}}_{\mathrm{rev}(R)} \cdot x^{D-\deg(R)}$$

Algorithm:	Complexity
1. Compute $A = 1/\mathrm{rev}(Q) \mod x^{D-d+1}$	$3\,\mathsf{M}(d) + O(d)$
2. Compute $\mathrm{rev}(U) = \mathrm{rev}(F) \cdot A \mod x^{D-d+1}$	$\mathsf{M}(d)$
3. Recover U and deduce $R = F - U \cdot Q$	$\mathsf{M}(d) + O(d)$

Fast polynomial division and modular exponentiation [Strassen, 1973]

Pb: Given $F \in \mathbb{K}[x]_{<2d}$ and $Q \in \mathbb{K}[x]_d$ compute (U, R) in Euclidean division

$$F = UQ + R$$

Naive algorithm: $O(d^2)$

Idea: when $\mathbb{K} = \mathbb{R}$, look at $F = UQ + R$ from infinity: $U \sim_{+\infty} F/Q$

Formalization: Let $D = \deg(F)$. Then $\deg(U) = D - d < d$, $\deg(R) < d$ and

$$\underbrace{F(1/x)x^D}_{\mathrm{rev}(F)} = \underbrace{Q(1/x)x^d}_{\mathrm{rev}(Q)} \cdot \underbrace{U(1/x)x^{D-d}}_{\mathrm{rev}(U)} + \underbrace{R(1/x)x^{\deg(R)}}_{\mathrm{rev}(R)} \cdot x^{D-\deg(R)}$$

Algorithm: Complexity

1. Compute $A = 1/\mathrm{rev}(Q) \mod x^{D-d+1}$ — $3\,\mathsf{M}(d) + O(d)$
2. Compute $\mathrm{rev}(U) = \mathrm{rev}(F) \cdot A \mod x^{D-d+1}$ — $\mathsf{M}(d)$
3. Recover U and deduce $R = F - U \cdot Q$ — $\mathsf{M}(d) + O(d)$

▷ Step 1 based on formal Newton iteration; it depends only on Q (not on F)

Fast polynomial division and modular exponentiation [Strassen, 1973]

Pb: Given $F \in \mathbb{K}[x]_{<2d}$ and $Q \in \mathbb{K}[x]_d$ compute (U, R) in Euclidean division

$$F = UQ + R$$

Naive algorithm: $O(d^2)$

Idea: when $\mathbb{K} = \mathbb{R}$, look at $F = UQ + R$ from infinity: $U \sim_{+\infty} F/Q$

Formalization: Let $D = \deg(F)$. Then $\deg(U) = D - d < d$, $\deg(R) < d$ and

$$\underbrace{F(1/x)x^D}_{\mathrm{rev}(F)} = \underbrace{Q(1/x)x^d}_{\mathrm{rev}(Q)} \cdot \underbrace{U(1/x)x^{D-d}}_{\mathrm{rev}(U)} + \underbrace{R(1/x)x^{\deg(R)}}_{\mathrm{rev}(R)} \cdot x^{D-\deg(R)}$$

Algorithm: Complexity

1. Compute $A = 1/\mathrm{rev}(Q) \mod x^{D-d+1}$ — $3\,\mathsf{M}(d) + O(d)$
2. Compute $\mathrm{rev}(U) = \mathrm{rev}(F) \cdot A \mod x^{D-d+1}$ — $\mathsf{M}(d)$
3. Recover U and deduce $R = F - U \cdot Q$ — $\mathsf{M}(d) + O(d)$

▷ Step 1 based on formal Newton iteration; it depends only on Q (not on F)

▷ Corollary: Modular exponentiation $x^N \bmod Q$ in $\sim 3\,\mathsf{M}(d) \log N$ ops. in $\mathbb{K}$

Application: fast modular exponentiation

Pb: Given $P, Q \in \mathbb{K}[x]_{<d}$ compute $P^N \bmod Q$

Application: fast modular exponentiation

Pb: Given $P, Q \in \mathbb{K}[x]_{<d}$ compute $P^N \bmod Q$

Naive algorithm: $O(Nd^2)$

Application: fast modular exponentiation

Pb: Given $P, Q \in \mathbb{K}[x]_{<d}$ compute $P^N \bmod Q$

Naive algorithm: $O(Nd^2)$

Better algorithm: binary powering in $R = \mathbb{K}[x]/(Q)$ $O\big(\log N\big)$ ops. in R

Application: fast modular exponentiation

Pb: Given $P, Q \in \mathbb{K}[x]_{<d}$ compute $P^N \bmod Q$

Naive algorithm: $O(Nd^2)$

Better algorithm: binary powering in $R = \mathbb{K}[x]/(Q)$ $O(\log N)$ ops. in R

Algorithm: Complexity

1. Precompute $A = 1/\mathrm{rev}(Q) \mod x^d$ $3\,\mathsf{M}(d) + O(d)$
2. Perform $\lfloor \log N \rfloor$ square-and-multiply modulo Q; for each $V^2 \bmod Q$:
   - compute the square $F := V^2$ $\mathsf{M}(d)$
   - compute the remainder $F \bmod Q$:
     - Compute $\mathrm{rev}(U) = \mathrm{rev}(F) \cdot A \mod x^d$ $\mathsf{M}(d)$
     - Recover U and deduce $R = F - U \cdot Q$ $\mathsf{M}(d) + O(d)$

Application: fast modular exponentiation

Pb: Given $P, Q \in \mathbb{K}[x]_{<d}$ compute $P^N \bmod Q$

Naive algorithm: $O(Nd^2)$

Better algorithm: binary powering in $R = \mathbb{K}[x]/(Q)$ $O(\log N)$ ops. in R

Algorithm: Complexity

1. Precompute $A = 1/\mathrm{rev}(Q) \mod x^d$ $3\,\mathsf{M}(d) + O(d)$
2. Perform $\lfloor \log N \rfloor$ square-and-multiply modulo Q; for each $V^2 \bmod Q$:
   - compute the square $F := V^2$ $\mathsf{M}(d)$
   - compute the remainder $F \bmod Q$:
     - Compute $\mathrm{rev}(U) = \mathrm{rev}(F) \cdot A \mod x^d$ $\mathsf{M}(d)$
     - Recover U and deduce $R = F - U \cdot Q$ $\mathsf{M}(d) + O(d)$

▷ $P^N \bmod Q$ in $3\,\mathsf{M}(d)\,(1 + \lfloor \log N \rfloor) + O(d \log N) \sim 3\,\mathsf{M}(d) \log N$ ops. in $\mathbb{K}$

Application: fast modular exponentiation

Pb: Given $P, Q \in \mathbb{K}[x]_{<d}$ compute $P^N \bmod Q$

Naive algorithm: $O(Nd^2)$

Better algorithm: binary powering in $R = \mathbb{K}[x]/(Q)$ $O(\log N)$ ops. in R

Algorithm: Complexity

1. Precompute $A = 1/\mathrm{rev}(Q) \mod x^d$ $3\,\mathsf{M}(d) + O(d)$
2. Perform $\lfloor \log N \rfloor$ square-and-multiply modulo Q; for each $V^2 \bmod Q$:
   - compute the square $F := V^2$ $\mathsf{M}(d)$
   - compute the remainder $F \bmod Q$:
     - Compute $\mathrm{rev}(U) = \mathrm{rev}(F) \cdot A \mod x^d$ $\mathsf{M}(d)$
     - Recover U and deduce $R = F - U \cdot Q$ $\mathsf{M}(d) + O(d)$

▷ $P^N \bmod Q$ in $3\,\mathsf{M}(d)\,(1 + \lfloor \log N \rfloor) + O(d \log N) \sim 3\,\mathsf{M}(d) \log N$ ops. in $\mathbb{K}$

▷ A bit optimistic (did not count "-and-multiply"...); OK if $P = x$

Schönhage's Golden Rules

RULE 2: *Do not waste a factor of two !*

Computing the N-th term of a C-recursive sequence

$$u_{n+d} = c_{d-1}u_{n+d-1} + \cdots + c_0 u_n, \qquad n \geq 0,$$

Computing the N-th term of a C-recursive sequence

$$u_{n+d} = c_{d-1}u_{n+d-1} + \cdots + c_0 u_n, \qquad n \geq 0,$$

rewrites

$$\underbrace{\begin{bmatrix} u_N \\ u_{N+1} \\ \vdots \\ u_{N+d-1} \end{bmatrix}}_{v_N} = \underbrace{\begin{bmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ c_0 & c_1 & \cdots & c_{d-1} \end{bmatrix}}_{C^T} \underbrace{\begin{bmatrix} u_{N-1} \\ u_N \\ \vdots \\ u_{N+d-2} \end{bmatrix}}_{v_{N-1}} = (C^T)^N \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{d-1} \end{bmatrix}}_{v_0}, \qquad N \geq 1.$$

Computing the N-th term of a C-recursive sequence

$$u_{n+d} = c_{d-1}u_{n+d-1} + \cdots + c_0 u_n, \qquad n \geq 0,$$

rewrites

$$\underbrace{\begin{bmatrix} u_N \\ u_{N+1} \\ \vdots \\ u_{N+d-1} \end{bmatrix}}_{v_N} = \underbrace{\begin{bmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ c_0 & c_1 & \cdots & c_{d-1} \end{bmatrix}}_{C^T} \underbrace{\begin{bmatrix} u_{N-1} \\ u_N \\ \vdots \\ u_{N+d-2} \end{bmatrix}}_{v_{N-1}} = (C^T)^N \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{d-1} \end{bmatrix}}_{v_0}, \qquad N \geq 1.$$

▷ [Miller, Spencer Brown, 1966]: binary powering for $(C^T)^N$ $\qquad O(d^\theta \log(N))$

Computing the N-th term of a C-recursive sequence

$$u_{n+d} = c_{d-1}u_{n+d-1} + \cdots + c_0 u_n, \qquad n \geq 0,$$

rewrites

$$\underbrace{\begin{bmatrix} u_N \\ u_{N+1} \\ \vdots \\ u_{N+d-1} \end{bmatrix}}_{v_N} = \underbrace{\begin{bmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ c_0 & c_1 & \cdots & c_{d-1} \end{bmatrix}}_{C^T} \underbrace{\begin{bmatrix} u_{N-1} \\ u_N \\ \vdots \\ u_{N+d-2} \end{bmatrix}}_{v_{N-1}} = (C^T)^N \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{d-1} \end{bmatrix}}_{v_0}, \qquad N \geq 1.$$

▷ [Miller, Spencer Brown, 1966]: binary powering for $(C^T)^N$ $\quad O(d^\theta \log(N))$

▷ [Fiduccia, 1985] binary powering in $\mathbb{K}[x]/(\Gamma)$, with $\Gamma = x^d - \sum_{i=0}^{d-1} c_i x^i$

$$u_N = e \cdot v_N = e \cdot (C^T)^N \cdot v_0 = \left(C^N \cdot e^T\right)^T \cdot v_0 = \langle x^N \bmod \Gamma, v_0 \rangle,$$

where $e = [1 \quad 0 \quad \cdots \quad 0]$. $\quad \sim 3\,\mathsf{M}(d) \log N$

Computing the N-th term of a C-recursive sequence

$$u_{n+d} = c_{d-1}u_{n+d-1} + \cdots + c_0 u_n, \qquad n \geq 0,$$

rewrites

$$\underbrace{\begin{bmatrix} u_N \\ u_{N+1} \\ \vdots \\ u_{N+d-1} \end{bmatrix}}_{v_N} = \underbrace{\begin{bmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ c_0 & c_1 & \cdots & c_{d-1} \end{bmatrix}}_{C^T} \underbrace{\begin{bmatrix} u_{N-1} \\ u_N \\ \vdots \\ u_{N+d-2} \end{bmatrix}}_{v_{N-1}} = (C^T)^N \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{d-1} \end{bmatrix}}_{v_0}, \qquad N \geq 1.$$

▷ [Miller, Spencer Brown, 1966]: binary powering for $(C^T)^N$ $\qquad O(d^\theta \log(N))$

▷ [Fiduccia, 1985] binary powering in $\mathbb{K}[x]/(\Gamma)$, with $\Gamma = x^d - \sum_{i=0}^{d-1} c_i x^i$

$$u_N = e \cdot v_N = e \cdot (C^T)^N \cdot v_0 = \left(C^N \cdot e^T\right)^T \cdot v_0 = \langle x^N \bmod \Gamma, v_0 \rangle,$$

where $e = [1 \quad 0 \quad \cdots \quad 0]$. $\qquad \sim 3\,\mathsf{M}(d) \log N$

▷ [B., Mori, 2020]: different ideas / algorithms $\qquad \sim 2\,\mathsf{M}(d) \log N$

Example: N-th term of the Fibonacci sequence

L'INTERMÉDIAIRE

DES

MATHÉMATICIENS

DIRIGÉ PAR

C.-A. LAISANT, DOCTEUR ÈS SCIENCES, | ÉMILE LEMOINE, INGÉNIEUR CIVIL,

ANCIENS ÉLÈVES DE L'ÉCOLE POLYTECHNIQUE.

TOME VI. — 1899.

PARIS,

GAUTHIER-VILLARS, IMPRIMEUR-LIBRAIRE

DU BUREAU DES LONGITUDES, DE L'ÉCOLE POLYTECHNIQUE,

55, Quai des Grands-Augustins, 55.

1899

Je serais également heureux d'avoir des renseignements bibliographiques (postérieurs à Delambre) sur les études *scientifiques* consacrées aux cadrans verticaux dans l'antiquité. PAUL TANNERY.

1539. **[H1b]** Il peut arriver qu'une équation différentielle admette une intégrale particulière imaginaire. La connaissance de celle-ci peut-elle être de quelque utilité pour l'intégration de l'équation donnée? H. BROCARD.

1540. **[I2b]** b étant un nombre composé, quelles sont les valeurs de b qui rendent le produit $1.2.3\ldots(b-1)$ non divisible par b? G. DE ROCQUIGNY.

1541. **[I25a]** Quel est le procédé le plus expéditif pour calculer un terme très éloigné dans la série de Fibonacci :

$$0,\quad 1,\quad 1,\quad 2,\quad 3,\quad 5,\quad 8,\quad \ldots?$$

G. DE ROCQUIGNY.

1542. **[E1a]** Est-il exact, et dans ce cas comment pourrait-on le démontrer, que :

1° L'expression

$$\Phi_n(x) = n^{x-1} - \frac{x}{1}(n-1)^{x-1} + \frac{x(x-1)}{1.2}(n-2)^{x-1} - \ldots,$$

où n désigne un entier et x une quantité positive quelconque, tend vers zéro en même temps que n vers l'infini;

2° La loi de décroissance des quantités $\Phi_n(x)$ est assez rapide pour que la série

$$\Phi_1(x) + \Phi_2(x) + \Phi_3(x) + \ldots + \Phi_n(x) + \ldots$$

soit convergente;

3° La somme de cette série a pour limite la fonction $\Gamma(x)$?

Tout cela étant, la fonction eulérienne de deuxième $\Gamma(1+x)$ se présenterait comme la limite de l'expression

$$n^x - \frac{x}{1}(n-1)^x + \frac{x(x-1)}{1.2}(n-2)^x - \ldots.$$

E.-A. Majol.

Example: N-th term of the Fibonacci sequence

Fiduccia's algorithm (1985): binary powering in the ring $\mathbb{K}[x]/(x^2 - x - 1)$:

$$C^n = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n = \text{ matrix of } \quad (x^n \bmod x^2 - x - 1)$$

$$\Longrightarrow \quad F_{n-2} + xF_{n-1} = \quad x^n \bmod x^2 - x - 1$$

Cost: $O(\log N)$ products in $\mathbb{K}[x]/(x^2 - x - 1) \longrightarrow O(\log N)$ ops. for F_N

Example: N-th term of the Fibonacci sequence

Fiduccia's algorithm (1985): binary powering in the ring $\mathbb{K}[x]/(x^2 - x - 1)$:

$$C^n = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n = \text{ matrix of } \quad (x^n \bmod x^2 - x - 1)$$

$$\implies \quad F_{n-2} + xF_{n-1} = \quad x^n \bmod x^2 - x - 1$$

Cost: $O(\log N)$ products in $\mathbb{K}[x]/(x^2 - x - 1) \longrightarrow O(\log N)$ ops. for F_N

Explains Shortt's algorithm (1978):

$$F_{2n-2} + xF_{2n-1} = (F_{n-2} + xF_{n-1})^2 \bmod x^2 - x - 1$$

Example: N-th term of the Fibonacci sequence

Fiduccia's algorithm (1985): binary powering in the ring $\mathbb{K}[x]/(x^2-x-1)$:

$$C^n = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n = \text{ matrix of } \quad (x^n \bmod x^2-x-1)$$

$$\Longrightarrow \quad F_{n-2} + xF_{n-1} = \quad x^n \bmod x^2-x-1$$

Cost: $O(\log N)$ products in $\mathbb{K}[x]/(x^2-x-1) \longrightarrow O(\log N)$ ops. for F_N

Explains Shortt's algorithm (1978):

$$F_{2n-2} + xF_{2n-1} = (F_{n-2}+xF_{n-1})^2 \bmod x^2-x-1$$

implies $\begin{cases} F_{2n-2} & = F_{n-2}^2 + F_{n-1}^2 \\ F_{2n-1} & = F_{n-1}^2 + 2F_{n-1}F_{n-2} \end{cases}$

$$(F_0, F_1) \to (F_2, F_3) \to (F_6, F_7) \to (F_{14}, F_{15}) \to \dots$$

Cost: $3\ \times$ and $3\ +$ per arrow

Example: N-th term of the Fibonacci sequence

L'INTERMÉDIAIRE

DES

MATHÉMATICIENS

DIRIGÉ PAR

C.-A. LAISANT, DOCTEUR ÈS SCIENCES,

ÉMILE LEMOINE, INGÉNIEUR CIVIL,

ANCIENS ÉLÈVES DE L'ÉCOLE POLYTECHNIQUE.

TOME VII. — 1900.

PARIS,

GAUTHIER-VILLARS, IMPRIMEUR-LIBRAIRE

DU BUREAU DES LONGITUDES, DE L'ÉCOLE POLYTECHNIQUE,

55, Quai des Grands-Augustins, 55.

1900

cise en prenant un plus grand nombre de décimales, nous paraît, en effet, l'un des procédés de calcul les meilleurs. Elle permet de trouver des termes, exactement, jusqu'à une limite assez éloignée; et pour des termes très lointains et impossibles à écrire, elle a le mérite de faire connaître le nombre des chiffres. C.-A. LAISANT.

La suite de Fibonacci peut être exprimée par une autre loi que celle qui est ordinairement employée. En effet, en considérant les termes de cette suite comme provenant du calcul des réduites d'une fraction continue périodique symétrique dont tous les quotients incomplets sont égaux à l'unité, on trouve, en appliquant les formules de Lucas (*Théorie des nombres*) :

$$u_{2n} = u_n^2 + u_{n-1}^2,$$
$$u_{2n-1} = u_{n-1}(u_n + u_{n-2}) = u_n^2 - u_{n-2}^2,$$

le point de départ étant

$$u_0 = 1, \qquad u_1 = 1, \qquad u_2 = 2.$$

Ces formules permettent de calculer assez rapidement les termes de cette suite. En moins d'un quart d'heure j'ai pu calculer

$$u_{100} = 573\,147\,844\,013\,817\,084\,101.$$

Accessoirement ces formules montrent que tous les termes de rang impair sont des nombres composés sauf le terme $u_3 = 3$ et que tous les nombres premiers, que l'on ne peut rencontrer qu'aux rangs pairs, sont tous de la forme $4u + 1$ puisqu'ils sont la somme de deux carrés premiers entre eux. G. PICOU.

Les valeurs de u_{100} donées par MM. *Rosace* et Picou sont toutes deus exactes, seulement M. *Rosace* prend pour premiers termes 0, 1, 1, 2, 3, ... et M. Picou 1, 1, 2, 3, ... E. LEMOINE.

1549. (**1899**, 150) (E. DUPORCQ). — *Sur une propriété nouvelle de l'ovale de Descartes.* — Soient R le rayon de courbure, θ, θ', θ'' les angles que font les rayons vecteurs avec la normale; selon que l'ovale est rapporté en coordonnées bipolaires aux foyers φ, φ'; φ, φ''; φ', φ'' on a pour le rayon de courbure les formes

$$(1) \quad R = \frac{\cos\theta + m\cos\theta'}{\frac{\cos^2\theta}{\rho} + m\frac{\cos^2\theta'}{\rho'}} = \frac{\cos\theta' + p\cos\theta''}{\frac{\cos^2\theta'}{\rho'} + p\frac{\cos^2\theta''}{\rho''}} = \frac{\cos\theta'' + q\cos\theta'}{\frac{\cos^2\theta''}{\rho''} + q\frac{\cos^2\theta'}{\rho'}}$$

Duality lemma (link between C-recursive sequences and rational functions)
Let $A(x) = \sum_{n\geq 0} u_n x^n \in \mathbb{K}[[x]]$ be the generating function of $(u_n)_{n\geq 0}$.
The following assertions are equivalent:

(i) $(u_n)_{n\geq 0}$ is C-recursive, having Γ as characteristic polynomial of degree d;

(ii) $A(x)$ is rational, of the form $A = P/Q$ for some $P \in \mathbb{K}[x]_{<d}$, where $Q := \mathsf{rev}_d(\Gamma) = \Gamma(\frac{1}{x})x^d$.

Duality lemma (link between C-recursive sequences and rational functions)
Let $A(x) = \sum_{n\geq 0} u_n x^n \in \mathbb{K}[[x]]$ be the generating function of $(u_n)_{n\geq 0}$.
The following assertions are equivalent:

(i) $(u_n)_{n\geq 0}$ is C-recursive, having Γ as characteristic polynomial of degree d;

(ii) $A(x)$ is rational, of the form $A = P/Q$ for some $P \in \mathbb{K}[x]_{<d}$, where $Q := \mathsf{rev}_d(\Gamma) = \Gamma(\frac{1}{x})x^d$.

▷ The denominator of A encodes a recurrence for $(u_n)_{n\geq 0}$; the numerator encodes initial conditions.

Computing the N-th coefficient of a rational function

Duality lemma (link between C-recursive sequences and rational functions)
Let $A(x) = \sum_{n \geq 0} u_n x^n \in \mathbb{K}[[x]]$ be the generating function of $(u_n)_{n \geq 0}$.
The following assertions are equivalent:

(i) $(u_n)_{n \geq 0}$ is C-recursive, having Γ as characteristic polynomial of degree d;

(ii) $A(x)$ is rational, of the form $A = P/Q$ for some $P \in \mathbb{K}[x]_{<d}$, where $Q := \mathsf{rev}_d(\Gamma) = \Gamma(\frac{1}{x})x^d$.

▷ The denominator of A encodes a recurrence for $(u_n)_{n \geq 0}$; the numerator encodes initial conditions.

▷ Generating function of $(F_n)_{n \geq 0}$ given by $F_0 = a, F_1 = b, F_{n+2} = F_{n+1} + F_n$ is $(a + (b-a)x)/(1 - x - x^2)$. Here $\Gamma = x^2 - x - 1$ and $P = a + (b-a)x$.

Duality lemma (link between C-recursive sequences and rational functions)
Let $A(x) = \sum_{n \geq 0} u_n x^n \in \mathbb{K}[[x]]$ be the generating function of $(u_n)_{n \geq 0}$.
The following assertions are equivalent:

(i) $(u_n)_{n \geq 0}$ is C-recursive, having Γ as characteristic polynomial of degree d;

(ii) $A(x)$ is rational, of the form $A = P/Q$ for some $P \in \mathbb{K}[x]_{<d}$, where $Q := \mathsf{rev}_d(\Gamma) = \Gamma(\frac{1}{x})x^d$.

▷ The denominator of A encodes a recurrence for $(u_n)_{n \geq 0}$; the numerator encodes initial conditions.

▷ Generating function of $(F_n)_{n \geq 0}$ given by $F_0 = a, F_1 = b, F_{n+2} = F_{n+1} + F_n$ is $(a + (b-a)x)/(1 - x - x^2)$. Here $\Gamma = x^2 - x - 1$ and $P = a + (b-a)x$.

▷ Corollary: N-th Taylor coeff. of $\frac{P}{Q} \in \mathbb{K}(x)_d$ in $\sim 3\,\mathsf{M}(d) \log N$ ops. in $\mathbb{K}$

BONUS: RECENT RESULTS

Computing the N-th coefficient of a rational function, revisited

Pb: Given $P, Q \in \mathbb{K}[x]$ with $\deg(P) < \deg(Q) =: d$ and $N \in \mathbb{N}$, compute

$$u_N = [x^N]\, \frac{P(x)}{Q(x)}$$

Computing the N-th coefficient of a rational function, revisited

Pb: Given $P, Q \in \mathbb{K}[x]$ with $\deg(P) < \deg(Q) =: d$ and $N \in \mathbb{N}$, compute

$$u_N = [x^N]\, \frac{P(x)}{Q(x)}$$

Idea: With $U(x) := P(x)Q(-x)$ and $V(x^2) := Q(x)Q(-x)$,

$$u_N = [x^N]\, \frac{P(x)Q(-x)}{Q(x)Q(-x)} = [x^N]\, \frac{U(x)}{V(x^2)}.$$

Computing the N-th coefficient of a rational function, revisited

Pb: Given $P, Q \in \mathbb{K}[x]$ with $\deg(P) < \deg(Q) =: d$ and $N \in \mathbb{N}$, compute

$$u_N = [x^N]\, \frac{P(x)}{Q(x)}$$

Idea: With $U(x) := P(x)Q(-x)$ and $V(x^2) := Q(x)Q(-x)$,

$$u_N = [x^N]\, \frac{P(x)Q(-x)}{Q(x)Q(-x)} = [x^N]\, \frac{U(x)}{V(x^2)}.$$

▷ Writing $U(x) = U_{\mathrm{e}}(x^2) + xU_{\mathrm{o}}(x^2)$, we have

$$u_N = \begin{cases} [x^N]\, \frac{U_{\mathrm{e}}(x^2)}{V(x^2)}, & \text{if } N \text{ is even} \\ [x^N]\, \frac{xU_{\mathrm{o}}(x^2)}{V(x^2)}, & \text{else.} \end{cases}$$

$$= \begin{cases} [x^{N/2}]\, \frac{U_{\mathrm{e}}(x)}{V(x)}, & \text{if } N \text{ is even} \\ [x^{(N-1)/2}]\, \frac{U_{\mathrm{o}}(x)}{V(x)}, & \text{else.} \end{cases}$$

Computing the N-th coefficient of a rational function, revisited

Pb: Given $P, Q \in \mathbb{K}[x]$ with $\deg(P) < \deg(Q) =: d$ and $N \in \mathbb{N}$, compute

$$u_N = [x^N]\, \frac{P(x)}{Q(x)}$$

Idea: With $U(x) := P(x)Q(-x)$ and $V(x^2) := Q(x)Q(-x)$,

$$u_N = [x^N]\, \frac{P(x)Q(-x)}{Q(x)Q(-x)} = [x^N]\, \frac{U(x)}{V(x^2)}.$$

▷ Writing $U(x) = U_{\mathrm{e}}(x^2) + xU_{\mathrm{o}}(x^2)$, we have

$$u_N = \begin{cases} [x^N]\, \frac{U_{\mathrm{e}}(x^2)}{V(x^2)}, & \text{if } N \text{ is even} \\ [x^N]\, \frac{xU_{\mathrm{o}}(x^2)}{V(x^2)}, & \text{else.} \end{cases}$$

$$= \begin{cases} [x^{N/2}]\, \frac{U_{\mathrm{e}}(x)}{V(x)}, & \text{if } N \text{ is even} \\ [x^{(N-1)/2}]\, \frac{U_{\mathrm{o}}(x)}{V(x)}, & \text{else.} \end{cases}$$

▷ Algorithm: repeat this reduction until $N \geq 1$ $\qquad 2\,\mathsf{M}(d)\lceil\log(N+1)\rceil$

Algorithm 1 OneCoeff

Input: $P(x)$, $Q(x)$, N

Output: $[x^N]\,\frac{P(x)}{Q(x)}$

Assumptions: $Q(0)$ invertible and $\deg(P) < \deg(Q) =: d$

```
while N ≥ 1 do
    U(x) ← P(x)Q(−x)                         ▷ U = ∑_{i=0}^{2d−1} U_i x^i
    if N is even then
        P(x) ← ∑_{i=0}^{d−1} U_{2i} x^i
    else
        P(x) ← ∑_{i=0}^{d−1} U_{2i+1} x^i
    end if
    A(x) ← Q(x)Q(−x)                         ▷ A = ∑_{i=0}^{2d} A_i x^i
    Q(x) ← ∑_{i=0}^{d} A_{2i} x^i
    N ← ⌊N/2⌋
end while
return P(0)/Q(0)
```

Computing the N-th term of a C-recursive sequence, revisited

Pb: Given $N \in \mathbb{N}$, $u_0, \ldots, u_{d-1} \in \mathbb{K}$, and the recurrence

$$u_{n+d} = c_{d-1}u_{n+d-1} + \cdots + c_0 u_n, \qquad n \geq 0,$$

compute u_N

Computing the N-th term of a C-recursive sequence, revisited

Pb: Given $N \in \mathbb{N}$, $u_0, \ldots, u_{d-1} \in \mathbb{K}$, and the recurrence

$$u_{n+d} = c_{d-1} u_{n+d-1} + \cdots + c_0 u_n, \qquad n \geq 0,$$

compute u_N

▷ [Fiduccia, 1985] binary powering in $\mathbb{K}[x]/(\Gamma)$, with $\Gamma = x^d - \sum_{i=0}^{d-1} c_i x^i$
$\sim 3\,\mathsf{M}(d) \log N$

Computing the N-th term of a C-recursive sequence, revisited

Pb: Given $N \in \mathbb{N}$, $u_0, \ldots, u_{d-1} \in \mathbb{K}$, and the recurrence

$$u_{n+d} = c_{d-1} u_{n+d-1} + \cdots + c_0 u_n, \qquad n \geq 0,$$

compute u_N

▷ [Fiduccia, 1985] binary powering in $\mathbb{K}[x]/(\Gamma)$, with $\Gamma = x^d - \sum_{i=0}^{d-1} c_i x^i$ $\sim 3\,\mathsf{M}(d) \log N$

▷ [B., Mori, 2020]: Use $[x^N]\, \frac{P(x)}{Q(x)}$ and duality lemma $\sim 2\,\mathsf{M}(d) \log N$

Computing the N-th term of a C-recursive sequence, revisited

Pb: Given $N \in \mathbb{N}$, $u_0, \ldots, u_{d-1} \in \mathbb{K}$, and the recurrence

$$u_{n+d} = c_{d-1} u_{n+d-1} + \cdots + c_0 u_n, \qquad n \geq 0,$$

compute u_N

▷ [Fiduccia, 1985] binary powering in $\mathbb{K}[x]/(\Gamma)$, with $\Gamma = x^d - \sum_{i=0}^{d-1} c_i x^i$
$\sim 3\,\mathsf{M}(d) \log N$

▷ [B., Mori, 2020]: Use $[x^N] \, \frac{P(x)}{Q(x)}$ and duality lemma $\sim 2\,\mathsf{M}(d) \log N$

▷ Appropriate choice is: $Q = \operatorname{rev}(\Gamma)$, and P such that $\frac{P}{Q} = \sum_{i=0}^{d-1} u_i x^i \bmod x^d$

Algorithm 2 OneTerm

Input: rec. $u_{n+d} = c_{d-1}u_{n+d-1} + \cdots + c_0 u_n,\ (n \geq 0)$, and $u_0, \ldots, u_{d-1}, N$
Output: u_N
Assumptions: $\Gamma(x) = x^d - \sum_{i=0}^{d-1} c_i x^i$ with $c_0 \neq 0$

1: $Q(x) \leftarrow x^d \Gamma(1/x)$
2: $P(x) \leftarrow (u_0 + \cdots + u_{d-1}x^{d-1}) \cdot Q(x) \bmod x^d$
3: **return** $[x^N]P(x)/Q(x)$ ▷ using Algorithm 1

▷ Advantage: faster than Fiduccia's algorithm

▷ in FFT-mode, $\sim \frac{2}{3}\,\mathsf{M}(d)\log N$ versus $\sim \frac{5}{3}\,\mathsf{M}(d)\log N$ [Shoup, NTL, 1995] and $\sim \frac{13}{12}\,\mathsf{M}(d)\log N$ [Mihăilescu, 2008]

▷ Drawback: computes a single u_N, while Fiduccia computes a whole slice

Application: new algorithm for the Fibonacci numbers

$$F_0 = 0, F_1 = 1, \quad F_{n+2} = F_{n+1} + F_n,\ n \geq 0.$$

Application: new algorithm for the Fibonacci numbers

$$F_0 = 0, F_1 = 1, \quad F_{n+2} = F_{n+1} + F_n, \; n \geq 0.$$

▷ Generating function $\sum_{n \geq 0} F_n x^n$ is $x/(1 - x - x^2)$.

Application: new algorithm for the Fibonacci numbers

$$F_0 = 0, F_1 = 1, \quad F_{n+2} = F_{n+1} + F_n,\ n \geq 0.$$

- ▷ Generating function $\sum_{n\geq 0} F_n x^n$ is $x/(1-x-x^2)$.
- ▷ Thus, the coefficient $F_N = [x^N]\ \frac{x}{1-x-x^2}$ is equal to

$$[x^N]\ \frac{x(1+x-x^2)}{1-3x^2+x^4} = \begin{cases} [x^{\frac{N}{2}}]\ \frac{x}{1-3x+x^2}, & \text{if } N \text{ is even,} \\ [x^{\frac{N-1}{2}}]\ \frac{1-x}{1-3x+x^2}, & \text{else.} \end{cases}$$

Application: new algorithm for the Fibonacci numbers

$$F_0 = 0, F_1 = 1, \quad F_{n+2} = F_{n+1} + F_n,\ n \geq 0.$$

- Generating function $\sum_{n\geq 0} F_n x^n$ is $x/(1-x-x^2)$.
- Thus, the coefficient $F_N = [x^N]\ \frac{x}{1-x-x^2}$ is equal to

$$[x^N]\ \frac{x(1+x-x^2)}{1-3x^2+x^4} = \begin{cases} [x^{\frac{N}{2}}]\ \frac{x}{1-3x+x^2}, & \text{if } N \text{ is even,} \\ [x^{\frac{N-1}{2}}]\ \frac{1-x}{1-3x+x^2}, & \text{else.} \end{cases}$$

- The computation of F_N is reduced to that of a coefficient of the form

$$[x^N]\ \frac{a+bx}{1-cx+x^2} = [x^N]\ \frac{(a+bx)(1+cx+x^2)}{1-(c^2-2)x^2+x^4}$$

which is equal to

$$\begin{cases} [x^{\frac{N}{2}}]\ \frac{a+(bc+a)x}{1-(c^2-2)x+x^2}, & \text{if } N \text{ is even,} \\ [x^{\frac{N-1}{2}}]\ \frac{(ac+b)+bx}{1-(c^2-2)x+x^2}, & \text{else.} \end{cases}$$

Algorithm 3 NewFibo

Input: N
Output: F_N
Assumptions: $N \geq 2$

```
c ← 3
if N is even then
    [a, b] ← [0, 1]
else
    [a, b] ← [1, −1]
end if
N ← ⌊N/2⌋
while N > 1 do
    if N is even then
        b ← a + b · c
    else
        a ← b + a · c
    end if
    c ← c^2 − 2
    N ← ⌊N/2⌋
end while
return b + a · c
```

Computation of $F_{43} = 433\,494\,437$ using the new algorithm

N	a	b	c
21	1	-1	3
10	$1 \times 3 - 1 = 2$		$3^2 - 2 = 7$
5		$(-1) \times 7 + 2 = -5$	$7^2 - 2 = 47$
2	$2 \times 47 - 5 = 89$		$47^2 - 2 = 2207$
1		$(-5) \times 2207 + 89$ $= -10946$	$2207^2 - 2 = 4870847$
0	$89 \times 4870847 - 10946$ $= 433494437$		

Algorithm 4 NewFiboPowerOfTwo

Input: N
Output: F_N
Assumptions: $N \geq 2$ and N is a power of 2

$[b, c] \leftarrow [1, 3]$
$N \leftarrow \lfloor N/2 \rfloor$
while $N > 2$ **do**
    $b \leftarrow b \cdot c$
    $c \leftarrow c^2 - 2$
    $N \leftarrow \lfloor N/2 \rfloor$
end while
return $b \cdot c$

▷ This is exactly [Cull, Holloway, 1989, Fig. 6], also [Knuth, 1969]

```
fib(n)
   f ← 1
   l ← 3
   for i = 2 to (log n − 1)
      f ← f * l
      l ← l * l − 2
   f ← f * l
   return f
```

Example: N-th term of the Fibonacci sequence

$u_{20} = 6765$; puis, pour $p = 20$, la même formule donnera

$$u_{100} = 354\,224\,848\,179\,261\,915\,075.$$

Les relations particulières qui précèdent pourraient être déduites directement de formules analogues à la formule (2), mais plus générales, telles que les suivantes :

$$u_{p+q+r-2} = u_{p-1}u_q u_r + u_p u_{q-1} u_r + u_p u_q u_{r-1} + u_{p-2} u_{q-2} u_{r-2},$$
$$u_{p+q+r-2} = u_p u_q u_r + u_p u_{q-1} u_{r-1} + u_{p-1} u_q u_{r-1} + u_{p-1} u_{q-1} u_r - u_{p-1} u_{q-1} u_{r-1};$$

Il suffit de faire $r = 1$ dans cette dernière pour retrouver la relation (2).

Rosace.

Soient les deux séries P_n (de Fibonacci) et Q_n, dont le $n^{\text{ième}}$ terme est au-dessous de l'indice n :

n	0	1	2	3	4	5	6	7	8	9	10
P_n	0	1	1	2	3	5	8	13	21	34	55
Q_n	2	1	3	4	7	11	18	29	47	76	123

La série Q_n procède des deux premiers termes 2, 1, de la même façon que la série P_n procède des termes 0 et 1. Les termes correspondants de ces deux séries présentent une remarquable analogie avec les fonctions trigonométriques *sinus* et *cosinus* :

$$P_n = \frac{1}{\sqrt{5}}[\alpha^n - (-1)^n \beta^n], \qquad Q_n = \alpha^n + (-1)^n \beta^n,$$

où $\alpha = \frac{\sqrt{5}+1}{2}$, $\beta = \frac{\sqrt{5}-1}{2}$.

De ces formules, on déduit les relations suivantes, correspondant à celles qui existent entre le *sinus* et le *cosinus* :

(1) $Q_n^2 - 5P_n^2 = 4(-1)^n,$

(2) $2P_{m+n} = P_m Q_n + Q_m P_n,$

(3) $2P_{m-n} = (-1)^n(P_m Q_n - Q_m P_n),$

(4) $P_{2m} = P_m Q_m,$

(5) $2Q_{m+n} = Q_m Q_n + 5P_m P_n,$

(6) $2Q_{m-n} = (-1)^n(Q_m Q_n - 5P_m P_n),$

(7) $P_{-m} = (-1)^{m+1} P_m.$

(8) $Q_{-m} = (-1)^m Q_m,$

(9) $2Q_{2m} = Q_m^2 + 5P_m^2,$

(10) $Q_{2m} = Q_m^2 - 2(-1)^m.$

Par combinaison de (2) et (3), nous avons

(11) $P_{m+n} = P_m Q_n - (-1)^n P_{m-n}.$

A l'aide de ces formules, notamment de (2), (4), (10) et (11), nous pourrons calculer très rapidement un terme P_n, d'une manière analogue à celle du sinus d'un arc multiple. Par exemple,

$$Q_{2^n} = 1, 3, 7, 47, \ldots,$$

chaque terme étant le carré du précédent, moins 2.

Pour une étude de ces séries et de quelques autres de ce genre, *voir* E. Lucas, *Théorie des fonctions numériques simplement périodiques* (*A. J. M.*, t. I, et *M. A. B.*, 1878).

E.-B. Escott (Grand Rapids, Mich.).

La série de Fibonacci, comme toute suite récurrente, est susceptible de deux modes de calcul : l'un le mode ordinaire, continu, qui n'est que l'application répétée de la formule même de récurrence; l'autre discontinu, auquel, dans le cas particulier envisagé, on est conduit par les considérations suivantes :

En partant des identités :

$$u_n + u_{n+1} - u_{n+2} = 0, \qquad \ldots, \qquad u_{n+m} + u_{n+m-1} - u_{n+m-1} = 0,$$

on a facilement $u_{n+m+1} = u_{n+1} u_{m+1} + u_n u_m$. Spécialement, pour $m = n$, j'aurai $u_{2n+1} = u_{n+1}^2 + u_n^2$ (égalité qui donne la décomposition en deux carrés d'un terme quelconque de rang impair de la série de Fibonacci), et de même, pour $m = n + 1$,

$$u_{2n+2} = u_{n+1}(u_{n+1} + 2u_n).$$

Ainsi les deux termes u_n et u_{n+1}, calculés par un moyen quelconque, suffisent à faire connaître les deux termes u_{2n+1} et u_{2n+2}, ceux-ci à faire connaître u_{4n+3} et u_{4n+4}, d'où l'on passera à u_{8n+7} et u_{8n+8}, etc. De proche en proche, on arrivera donc aux termes d'indices $(n+1)2^k - 1$ et $(n+1)2^k$, où n et k représentent des entiers à notre choix. Pour fixer les idées, soit à calculer la valeur numérique du $900^{\text{ième}}$ terme de la série. Le nombre 900 est égal à $7 \times 2^7 + 4$, j'aurai à chercher, de la façon qui vient d'être indiquée, u_{895} et u_{896}, puis, par le moyen de l'échelle de relation, je passerai successivement aux termes u_{897}, u_{898}, u_{899} et enfin u_{900}. Ainsi, les valeurs $u_{13} = 233$, $u_{14} = 377$ étant prises comme point de départ, j'en tirerai $u_{27} = 196418$, $u_{28} = 317811$, puis

$$u_{55} = 139583862445, \qquad u_{56} = 225851433717.$$

Schönhage's Golden Rules

Rule 8: *The development of fast algorithms is slow !*