N-th term of a linear recurrent sequence

Alin Bostan

Inria

ENS Lyon

M2, CR11

October 21, 2019

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.

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

$$z' = 2yy',$$

$$z'' = 2y'^2 + 2yy'' = 2y'^2 + \frac{2x}{1-x^2} yy',$$

$$z''' = 4y'y'' + \frac{2x}{1-x^2}(y'^2 + yy'') + \left(\frac{2}{1-x^2} + \frac{4x^2}{(1-x^2)^2}\right) yy'$$

$$= \left(\frac{2}{1-x^2} + \frac{6x^2}{(1-x^2)^2}\right) yy' + \frac{6x}{1-x^2} y'^2.$$

Solution, Part 2.

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

$$z' = 2yy',$$

$$z'' = 2y'^2 + 2yy'' = 2y'^2 + \frac{2x}{1-x^2} yy',$$

$$z''' = 4y'y'' + \frac{2x}{1-x^2}(y'^2 + yy'') + \left(\frac{2}{1-x^2} + \frac{4x^2}{(1-x^2)^2}\right) yy'$$

$$= \left(\frac{2}{1-x^2} + \frac{6x^2}{(1-x^2)^2}\right) yy' + \frac{6x}{1-x^2} y'^2.$$

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

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

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

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 $y'' = \frac{x}{1-x^2} y'$.

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

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

▷ $z = \sum_n a_n x^n$ 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

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

BINARY POWERING

Linear recurrences with constant coefficients

Def. $(a_n)_{n\geq 0}$ is a linearly recurrent sequence with constant coefficients (l.r.s.c.c, or C-recursive) if there exist $c_0, \ldots, c_{d-1} \in \mathbb{K}$ such that

$$a_{n+d} = c_{d-1}a_{n+d-1} + \cdots + c_0 a_n, \quad \text{for all } n \geq 0.$$

Linear recurrences with constant coefficients

Def. $(a_n)_{n\geq 0}$ is a linearly recurrent sequence with constant coefficients (l.r.s.c.c, or C-recursive) if there exist $c_0, \dots, c_{d-1} \in \mathbb{K}$ such that

$$a_{n+d} = c_{d-1}a_{n+d-1} + \cdots + c_0 a_n, \quad \text{for all } n \geq 0.$$

▷ $x^d - c_{d-1}x^{d-1} - \cdots - c_0$ is called a characteristic polynomial of $(a_n)_{n\geq 0}$.

Def. $(a_n)_{n\geq 0}$ is a linearly recurrent sequence with constant coefficients (l.r.s.c.c, or C-recursive) if there exist $c_0, \ldots, c_{d-1} \in \mathbb{K}$ such that

$$a_{n+d} = c_{d-1}a_{n+d-1} + \cdots + c_0 a_n, \quad \text{for all } n \geq 0.$$

▷ $x^d - c_{d-1}x^{d-1} - \cdots - c_0$ is called a characteristic polynomial of $(a_n)_{n\geq 0}$.

▷ The minimal polynomial of $(a_n)_{n\geq 0}$, denoted $\mathsf{MinPol}(a_n)$, is the polynomial of minimal degree among all characteristic polynomials of $(a_n)_{n\geq 0}$.

Def. $(a_n)_{n\geq 0}$ is a linearly recurrent sequence with constant coefficients (l.r.s.c.c, or C-recursive) if there exist $c_0, \ldots, c_{d-1} \in \mathbb{K}$ such that

$$a_{n+d} = c_{d-1}a_{n+d-1} + \cdots + c_0 a_n, \quad \text{for all } n \geq 0.$$

▷ $x^d - c_{d-1}x^{d-1} - \cdots - c_0$ is called a characteristic polynomial of $(a_n)_{n\geq 0}$.

▷ The minimal polynomial of $(a_n)_{n\geq 0}$, denoted $\mathsf{MinPol}(a_n)$, is the polynomial of minimal degree among all characteristic polynomials of $(a_n)_{n\geq 0}$.

▷ E.g., the Fibonacci seq. $(F_n)_{n\geq 0}$ given by $F_0 = F_1 = 1$, $F_{n+2} = F_{n+1} + F_n$ is a l.r.s.c.c., with $\mathsf{MinPol}(F_n) = x^2 - x - 1$. A char. poly is $x^3 + \frac{1}{2}x^2 - \frac{5}{2}x - \frac{3}{2}$.

Def. $(a_n)_{n\geq 0}$ is a linearly recurrent sequence with constant coefficients (l.r.s.c.c, or C-recursive) if there exist $c_0, \dots, c_{d-1} \in \mathbb{K}$ such that

$$a_{n+d} = c_{d-1}a_{n+d-1} + \cdots + c_0 a_n, \quad \text{for all } n \geq 0.$$

▷ $x^d - c_{d-1}x^{d-1} - \cdots - c_0$ is called a characteristic polynomial of $(a_n)_{n\geq 0}$.

▷ The minimal polynomial of $(a_n)_{n\geq 0}$, denoted $\mathrm{MinPol}(a_n)$, is the polynomial of minimal degree among all characteristic polynomials of $(a_n)_{n\geq 0}$.

▷ E.g., the Fibonacci seq. $(F_n)_{n\geq 0}$ given by $F_0 = F_1 = 1$, $F_{n+2} = F_{n+1} + F_n$ is a l.r.s.c.c., with $\mathrm{MinPol}(F_n) = x^2 - x - 1$. A char. poly is $x^3 + \frac{1}{2}x^2 - \frac{5}{2}x - \frac{3}{2}$.

▷ Central question today: how fast can one compute N-th coefficient a_N?

Def. $(a_n)_{n\geq 0}$ is a linearly recurrent sequence with constant coefficients (l.r.s.c.c, or C-recursive) if there exist $c_0, \ldots, c_{d-1} \in \mathbb{K}$ such that

$$a_{n+d} = c_{d-1}a_{n+d-1} + \cdots + c_0 a_n, \quad \text{for all } n \geq 0.$$

▷ $x^d - c_{d-1}x^{d-1} - \cdots - c_0$ is called a characteristic polynomial of $(a_n)_{n\geq 0}$.

▷ The minimal polynomial of $(a_n)_{n\geq 0}$, denoted $\mathsf{MinPol}(a_n)$, is the polynomial of minimal degree among all characteristic polynomials of $(a_n)_{n\geq 0}$.

▷ E.g., the Fibonacci seq. $(F_n)_{n\geq 0}$ given by $F_0 = F_1 = 1$, $F_{n+2} = F_{n+1} + F_n$ is a l.r.s.c.c., with $\mathsf{MinPol}(F_n) = x^2 - x - 1$. A char. poly is $x^3 + \frac{1}{2}x^2 - \frac{5}{2}x - \frac{3}{2}$.

▷ Central question today: how fast can one compute N-th coefficient a_N?

Same question if, more generally, $(a_n)_{n\geq 0}$ is P-recursive.

Binary powering

Problem: Given a ring $\mathbb{A}$, $a \in \mathbb{A}$ and $N \geq 1$, compute a^N

Binary powering

Problem: Given a ring $\mathbb{A}$, $a \in \mathbb{A}$ and $N \geq 1$, compute a^N

▷ Naive (iterative) algorithm: $O(N)$ ops. in $\mathbb{A}$

Binary powering

Problem: Given a ring $\mathbb{A}$, $a \in \mathbb{A}$ and $N \geq 1$, compute a^N

▷ Naive (iterative) algorithm: $O(N)$ ops. in $\mathbb{A}$

▷ Better algorithm (Pingala, 200 BC): $O(\log N)$ ops. in $\mathbb{A}$

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

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

Particular case: Modular exponentiation

- $\mathbb{A} = \mathbb{Z}/A\mathbb{Z}$ $\{+, \times\}$ in $\mathbb{A}$ have cost $O(\mathsf{M}_{\mathbb{Z}}(\log A))$ bit ops.

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

- $\mathbb{A} = \mathbb{K}[x]/(P)$ $\{+, \times\}$ in $\mathbb{A}$ have cost $O(\mathsf{M}(\deg P))$ ops. in $\mathbb{K}$

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

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;
```

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

Example: N-th term of the Fibonacci sequence

Problem: Compute F_N in $\mathbb{K}$, where

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

Example: N-th term of the Fibonacci sequence

Problem: Compute F_N in $\mathbb{K}$, where

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

▷ Naive (iterative) algorithm: N ops. in $\mathbb{K}$ to compute F_N

Problem: Compute F_N in $\mathbb{K}$, where

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

▷ Naive (iterative) algorithm: N ops. in $\mathbb{K}$ to compute F_N

▷ Folkore trick:

$$\begin{bmatrix} F_N \\ F_{N+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}}_{C} \begin{bmatrix} F_{N-1} \\ F_N \end{bmatrix} = C^N \begin{bmatrix} F_0 \\ F_1 \end{bmatrix}, \qquad N \geq 1.$$

Example: N-th term of the Fibonacci sequence

Problem: Compute F_N in $\mathbb{K}$, where

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

▷ Naive (iterative) algorithm: N ops. in $\mathbb{K}$ to compute F_N

▷ Folkore trick:

$$\begin{bmatrix} F_N \\ F_{N+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}}_{C} \begin{bmatrix} F_{N-1} \\ F_N \end{bmatrix} = C^N \begin{bmatrix} F_0 \\ F_1 \end{bmatrix}, \qquad N \geq 1.$$

▷ Binary powering: compute C^N recursively, using

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

Cost: $O(\log N)$ products of 2×2 matrices $\longrightarrow$ $O(\log N)$ ops. in $\mathbb{K}$ for F_N.

Example: N-th term of the Fibonacci sequence

$$\begin{bmatrix} F_{n-2} & F_{n-1} \\ F_{n-1} & F_n \end{bmatrix} = C^n \quad \Longrightarrow \quad \begin{bmatrix} F_{2n-2} & F_{2n-1} \\ F_{2n-1} & F_{2n} \end{bmatrix} = \begin{bmatrix} F_{n-2} & F_{n-1} \\ F_{n-1} & F_n \end{bmatrix}^2$$

Example: N-th term of the Fibonacci sequence

$$\begin{bmatrix} F_{n-2} & F_{n-1} \\ F_{n-1} & F_n \end{bmatrix} = C^n \quad \Longrightarrow \quad \begin{bmatrix} F_{2n-2} & F_{2n-1} \\ F_{2n-1} & F_{2n} \end{bmatrix} = \begin{bmatrix} F_{n-2} & F_{n-1} \\ F_{n-1} & F_n \end{bmatrix}^2$$

▷ The previous algorithm computes (by squaring 2×2 symmetric matrices)

$$(F_0, F_1, F_2) \to (F_2, F_3, F_4) \to (F_6, F_7, F_8) \to (F_{14}, F_{15}, F_{16}) \to \dots$$

Cost: $5\ \times$ and $3\ +$ per arrow

$$\begin{bmatrix} F_{n-2} & F_{n-1} \\ F_{n-1} & F_n \end{bmatrix} = C^n \quad \Longrightarrow \quad \begin{bmatrix} F_{2n-2} & F_{2n-1} \\ F_{2n-1} & F_{2n} \end{bmatrix} = \begin{bmatrix} F_{n-2} & F_{n-1} \\ F_{n-1} & F_n \end{bmatrix}^2$$

▷ The previous algorithm computes (by squaring 2×2 symmetric matrices)

$$(F_0, F_1, F_2) \to (F_2, F_3, F_4) \to (F_6, F_7, F_8) \to (F_{14}, F_{15}, F_{16}) \to \dots$$

Cost: $5\ \times$ and $3\ +$ per arrow

▷ Shortt's variant (1978): uses $\begin{cases} F_{2n-2} &= F_{n-2}^2 + F_{n-1}^2 \\ F_{2n-1} &= F_{n-1}^2 + 2F_{n-1}F_{n-2} \end{cases}$ and computes

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

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

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

$$\begin{aligned} C^n = &\quad \text{matrix of} \quad (x^n \bmod x^2 - x - 1) \\ \Longrightarrow \quad F_{n-2} + xF_{n-1} = &\quad x^n \bmod x^2 - x - 1 \end{aligned}$$

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 $\mathbb{K}[x]/(x^2-x-1)$:

$$\begin{aligned} C^n &= \text{ matrix of } \quad (x^n \bmod x^2-x-1) \\ \Longrightarrow \quad F_{n-2}+xF_{n-1} &= x^n \bmod x^2-x-1 \end{aligned}$$

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:

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

N-th term of a C-recursive sequence, general case

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

N-th term of a C-recursive sequence, general case

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

rewrites

$$\underbrace{\begin{bmatrix} a_N \\ a_{N+1} \\ \vdots \\ a_{N+d-1} \end{bmatrix}}_{v_N} = \underbrace{\begin{bmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ c_0 & c_1 & \cdots & c_{d-1} \end{bmatrix}}_{C^T} \underbrace{\begin{bmatrix} a_{N-1} \\ a_N \\ \vdots \\ a_{N+d-2} \end{bmatrix}}_{v_{N-1}} = (C^T)^N \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{d-1} \end{bmatrix}}_{v_0}, \qquad N \geq 1.$$

N-th term of a C-recursive sequence, general case

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

rewrites

$$\underbrace{\begin{bmatrix} a_N \\ a_{N+1} \\ \vdots \\ a_{N+d-1} \end{bmatrix}}_{v_N} = \underbrace{\begin{bmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ c_0 & c_1 & \cdots & c_{d-1} \end{bmatrix}}_{C^T} \underbrace{\begin{bmatrix} a_{N-1} \\ a_N \\ \vdots \\ a_{N+d-2} \end{bmatrix}}_{v_{N-1}} = (C^T)^N \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{d-1} \end{bmatrix}}_{v_0}, \qquad N \geq 1.$$

▷ Folklore trick: compute $(C^T)^N$ by binary powering $\qquad O(d^\omega \log(N))$

N-th term of a C-recursive sequence, general case

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

rewrites

$$\underbrace{\begin{bmatrix} a_N \\ a_{N+1} \\ \vdots \\ a_{N+d-1} \end{bmatrix}}_{v_N} = \underbrace{\begin{bmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ c_0 & c_1 & \cdots & c_{d-1} \end{bmatrix}}_{C^T} \underbrace{\begin{bmatrix} a_{N-1} \\ a_N \\ \vdots \\ a_{N+d-2} \end{bmatrix}}_{v_{N-1}} = (C^T)^N \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{d-1} \end{bmatrix}}_{v_0}, \qquad N \geq 1.$$

▷ Folklore trick: compute $(C^T)^N$ by binary powering $O(d^\omega \log(N))$

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

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

where $e = [1 \quad 0 \quad \cdots \quad 0]$.

N-th term of a C-recursive sequence, general case

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

rewrites

$$\underbrace{\begin{bmatrix} a_N \\ a_{N+1} \\ \vdots \\ a_{N+d-1} \end{bmatrix}}_{v_N} = \underbrace{\begin{bmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ c_0 & c_1 & \cdots & c_{d-1} \end{bmatrix}}_{C^T} \underbrace{\begin{bmatrix} a_{N-1} \\ a_N \\ \vdots \\ a_{N+d-2} \end{bmatrix}}_{v_{N-1}} = (C^T)^N \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{d-1} \end{bmatrix}}_{v_0}, \qquad N \geq 1.$$

▷ Folklore trick: compute $(C^T)^N$ by binary powering $O(d^\omega \log(N))$

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

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

where $e = [1 \quad 0 \quad \cdots \quad 0]$.

Cost: $O(\log N)$ products in $\mathbb{K}[x]/(P)$ $O(\mathsf{M}(d) \log N)$

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

(i) $(a_n)_{n\geq 0}$ is a l.r.s.c.c., having P as characteristic polynomial of degree d;

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

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

(i) $(a_n)_{n\geq 0}$ is a l.r.s.c.c., having P as characteristic polynomial of degree d;

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

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

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

(i) $(a_n)_{n\geq 0}$ is a l.r.s.c.c., having P as characteristic polynomial of degree d;

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

▷ The denominator of A encodes a recurrence for $(a_n)_{n\geq 0}$; 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 $P = x^2 - x - 1$ and $Q = a + (b-a)x$.

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

(i) $(a_n)_{n\geq 0}$ is a l.r.s.c.c., having P as characteristic polynomial of degree d;

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

▷ The denominator of A encodes a recurrence for $(a_n)_{n\geq 0}$; 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 $P = x^2 - x - 1$ and $Q = a + (b-a)x$.

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

Question[†]: The number of ways one can change any amount of banknotes of $10\,€, 20\,€, \ldots$ using coins of 50 cents, $1\,€$ and $2\,€$ is always a perfect square.

[†] Inspired by Pb. 1, Ch. 1, p. 1, vol. 1 of Pólya and Szegö's Problems Book (1925).

N-th term, with N as a parameter

Question[†]: The number of ways one can change any amount of banknotes of $10\,€, 20\,€, \ldots$ using coins of 50 cents, $1\,€$ and $2\,€$ is always a perfect square.

▷ This is equivalent to finding the number M_{20k} of solutions $(a, b, c) \in \mathbb{N}^3$ of

$$a + 2b + 4c = 20k.$$

† Inspired by Pb. 1, Ch. 1, p. 1, vol. 1 of Pólya and Szegö's Problems Book (1925).

N-th term, with N as a parameter

▷ Euler-Comtet's denumerants: $$\sum_{n \geq 0} M_n x^n = \frac{1}{(1-x)(1-x^2)(1-x^4)}.$$

N-th term, with N as a parameter

▷ Euler-Comtet's denumerants: $$\sum_{n\geq 0} M_n x^n = \frac{1}{(1-x)(1-x^2)(1-x^4)}.$$

```
> f:=1/(1-x)/(1-x^2)/(1-x^4):
> S:=series(f,x,201):
> [seq(coeff(S,x,20*k),k=1..10)];
```

[36, 121, 256, 441, 676, 961, 1296, 1681, 2116, 2601]

N-th term, with N as a parameter

▷ Euler-Comtet's denumerants: $\displaystyle\sum_{n\geq 0} M_n x^n = \frac{1}{(1-x)(1-x^2)(1-x^4)}.$

```
> f:=1/(1-x)/(1-x^2)/(1-x^4):
> S:=series(f,x,201):
> [seq(coeff(S,x,20*k),k=1..10)];
```

$$[36, 121, 256, 441, 676, 961, 1296, 1681, 2116, 2601]$$

```
> subs(n=20*k,gfun[ratpolytocoeff](f,x,n));
```

$$\frac{17}{32} + \frac{(20k+1)(20k+2)}{16} + 5k + \frac{(-1)^{-20k}(20k+1)}{16} + \frac{5(-1)^{-20k}}{32} + \sum_{\alpha_1^2+1=0} \left(- \frac{(\frac{1}{16} - \frac{1}{16}\alpha_1)\alpha_1^{-20k}}{\alpha_1} \right)$$

N-th term, with N as a parameter

▷ Euler-Comtet's denumerants: $$\sum_{n\geq 0} M_n x^n = \frac{1}{(1-x)(1-x^2)(1-x^4)}.$$

```
> f:=1/(1-x)/(1-x^2)/(1-x^4):
> S:=series(f,x,201):
> [seq(coeff(S,x,20*k),k=1..10)];
```

$$[36, 121, 256, 441, 676, 961, 1296, 1681, 2116, 2601]$$

```
> subs(n=20*k,gfun[ratpolytocoeff](f,x,n));
```

$$\frac{17}{32} + \frac{(20k+1)(20k+2)}{16} + 5k + \frac{(-1)^{-20k}(20k+1)}{16} + \frac{5(-1)^{-20k}}{32} + \sum_{\alpha_1^2+1=0}\left(-\frac{(\frac{1}{16}-\frac{1}{16}\alpha_1)\alpha_1^{-20k}}{\alpha_1}\right)$$

```
> value(subs(_alpha1^(-20*k)=1,%)):
> simplify(%) assuming k::posint:
> factor(%);
```

$$(5k+1)^2$$

BINARY SPLITTING

Example: fast factorial

Problem: Compute $N! = 1 \times \cdots \times N$

Example: fast factorial

Problem: Compute $N! = 1 \times \cdots \times N$

Naive (iterative) algorithm: unbalanced multiplicands $\tilde{O}(N^2)$

Example: fast factorial

Problem: Compute $N! = 1 \times \cdots \times N$

Naive (iterative) algorithm: unbalanced multiplicands $\tilde{O}(N^2)$

- Binary Splitting: balance computation sequence so as to take advantage of fast multiplication (operands of same sizes):

$$N! = \underbrace{(1 \times \cdots \times \lfloor N/2 \rfloor)}_{\text{size } \frac{1}{2} N \log N} \times \underbrace{((\lfloor N/2 \rfloor + 1) \times \cdots \times N)}_{\text{size } \frac{1}{2} N \log N}$$

and recurse. Complexity $\tilde{O}(N)$.

Example: fast factorial

Problem: Compute $N! = 1 \times \cdots \times N$

Naive (iterative) algorithm: unbalanced multiplicands $\tilde{O}(N^2)$

- Binary Splitting: balance computation sequence so as to take advantage of fast multiplication (operands of same sizes):

$$N! = \underbrace{(1 \times \cdots \times \lfloor N/2 \rfloor)}_{\text{size } \frac{1}{2} N \log N} \times \underbrace{((\lfloor N/2 \rfloor + 1) \times \cdots \times N)}_{\text{size } \frac{1}{2} N \log N}$$

  and recurse. Complexity $\tilde{O}(N)$.

- Extends to matrix factorials $A(N)A(N-1)\cdots A(1)$ $\tilde{O}(N)$
  $\longrightarrow$ recurrences of arbitrary order.

Problem: Compute the N-th term u_N of a P-recursive sequence

$$p_r(n)u_{n+r} + \cdots + p_0(n)u_n = 0, \qquad (n \in \mathbb{N})$$

Application to recurrences

Problem: Compute the N-th term u_N of a P-recursive sequence

$$p_r(n)u_{n+r} + \cdots + p_0(n)u_n = 0, \qquad (n \in \mathbb{N})$$

Naive algorithm: unroll the recurrence $\tilde{O}(N^2)$ bit ops.

Application to recurrences

Problem: Compute the N-th term u_N of a P-recursive sequence

$$p_r(n)u_{n+r} + \cdots + p_0(n)u_n = 0, \qquad (n \in \mathbb{N})$$

Naive algorithm: unroll the recurrence $\tilde{O}(N^2)$ bit ops.

Binary splitting: $U_n = (u_n, \ldots, u_{n+r-1})^T$ satisfies the 1st order recurrence

$$U_{n+1} = \frac{1}{p_r(n)} A(n) U_n \quad \text{with} \quad A(n) = \begin{bmatrix} & p_r(n) & & \\ & & \ddots & \\ & & & p_r(n) \\ -p_0(n) & -p_1(n) & \ldots & -p_{r-1}(n) \end{bmatrix}.$$

Problem: Compute the N-th term u_N of a P-recursive sequence

$$p_r(n)u_{n+r} + \cdots + p_0(n)u_n = 0, \qquad (n \in \mathbb{N})$$

Naive algorithm: unroll the recurrence $\tilde{O}(N^2)$ bit ops.

Binary splitting: $U_n = (u_n, \ldots, u_{n+r-1})^T$ satisfies the 1st order recurrence

$$U_{n+1} = \frac{1}{p_r(n)} A(n) U_n \quad \text{with} \quad A(n) = \begin{bmatrix} & p_r(n) & & \\ & & \ddots & \\ & & & p_r(n) \\ -p_0(n) & -p_1(n) & \ldots & -p_{r-1}(n) \end{bmatrix}.$$

$\Longrightarrow u_N$ reads off the matrix factorial $A(N-1) \cdots A(0)$

Problem: Compute the N-th term u_N of a P-recursive sequence

$$p_r(n)u_{n+r} + \cdots + p_0(n)u_n = 0, \qquad (n \in \mathbb{N})$$

Naive algorithm: unroll the recurrence $\tilde{O}(N^2)$ bit ops.

Binary splitting: $U_n = (u_n, \ldots, u_{n+r-1})^T$ satisfies the 1st order recurrence

$$U_{n+1} = \frac{1}{p_r(n)} A(n) U_n \quad \text{with} \quad A(n) = \begin{bmatrix} & p_r(n) & & \\ & & \ddots & \\ & & & p_r(n) \\ -p_0(n) & -p_1(n) & \ldots & -p_{r-1}(n) \end{bmatrix}.$$

$\Longrightarrow u_N$ reads off the matrix factorial $A(N-1)\cdots A(0)$

[Chudnovsky-Chudnovsky, 1987]: Binary splitting strategy $\tilde{O}(N)$ bit ops.

Application: fast computation of $e = \exp(1)$ [Brent 1976]

$$e_n = \sum_{k=0}^{n} \frac{1}{k!} \quad \longrightarrow \quad \exp(1) = 2.7182818284590452\ldots$$

Recurrence $e_n - e_{n-1} = 1/n! \iff n(e_n - e_{n-1}) = e_{n-1} - e_{n-2}$ rewrites

$$\begin{bmatrix} e_{N-1} \\ e_N \end{bmatrix} = \frac{1}{N} \underbrace{\begin{bmatrix} 0 & N \\ -1 & N+1 \end{bmatrix}}_{C(N)} \begin{bmatrix} e_{N-2} \\ e_{N-1} \end{bmatrix} = \frac{1}{N!} C(N)C(N-1)\cdots C(1) \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

▷ e_N in $\tilde{O}(N)$ bit operations [Brent 1976]

▷ generalizes to the evaluation of any D-finite series at an algebraic number [Chudnovsky-Chudnovsky 1987] $\tilde{O}(N)$ bit ops.

Implementation in gfun [Mezzarobba, Salvy 2010]

```
> rec:={n*(e(n) - e(n-1)) = e(n-1) - e(n-2), e(0)=1, e(1)=2};
> pro:=rectoproc(rec,e(n));
```

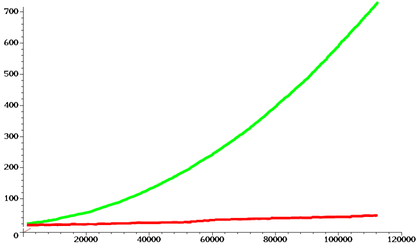
```
pro := proc(n::nonnegint)
local i1, loc0, loc1, loc2, tmp2, tmp1, i2;
  if n <= 22 then
    loc0 := 1;  loc1 := 2;
    if n = 0 then return loc0
      else for i1 to n - 1 do
         loc2 := (-loc0 + loc1 + loc1*(i1 + 1))/(i1 + 1);
         loc0 := loc1; loc1 := loc2
      end do
    end if; loc1
  else
  tmp1 := `gfun/rectoproc/binsplit`([
    'ndmatrix'(Matrix([[0, i2 + 2], [-1, i2 + 3]]), i2 + 2), i2, 0, n,
     matrix_ring(ad, pr, ze, ndmatrix(Matrix(2, 2, [[...],[...]],
     datatype = anything, storage = empty, shape = [identity]), 1)),
     expected_entry_size], Vector(2, [...], datatype = anything));
  tmp1 := subs({e(0) = 1, e(1) = 2}, tmp1);  tmp1
  end if
end proc
```

```
> tt:=time(): x:=pro(210000): time()-tt;
> tt:=time(): y:=evalf(exp(1), 1000000): time()-tt, evalf(x-y, 1000000);
```

3.730, 24.037, 0.

[Chudnovsky-Chudnovsky 1987] fast convergence hypergeometric identity

$$\frac{1}{\pi} = \frac{1}{53360\sqrt{640320}} \sum_{n\geq 0} \frac{(-1)^n (6n)!(13591409 + 545140134n)}{n!^3(3n)!(8 \cdot 100100025 \cdot 327843840)^n}.$$



▷ Used in Maple & Mathematica: 1st order recurrence, yields 14 correct digits per iteration $\longrightarrow$ 4 billion digits [Chudnovsky-Chudnovsky 1994]

▷ Current record: 31.4 trillion digits [Iwao 2019]

Example [Flajolet, Salvy, 1997]

What is the coefficient of x^{3000} in the expansion of

$$(x+1)^{2000}\left(x^2+x+1\right)^{1000}\left(x^4+x^3+x^2+x+1\right)^{500}$$

Example [Flajolet, Salvy, 1997]

What is the coefficient of x^{3000} in the expansion of

$$(x+1)^{2000}\left(x^2+x+1\right)^{1000}\left(x^4+x^3+x^2+x+1\right)^{500}$$

```
> st:=time(); n1:=2000; n2:=1000; n3:=500;
> P1:=x+1; P2:=x^2+x+1; P3:=x^4+x^3+x^2+x+1;
> d1:={diff(u(x),x)*P1-n1*diff(P1,x)*u(x)=0, u(0)=1}:
> d2:={diff(v(x),x)*P2-n2*diff(P2,x)*v(x)=0, v(0)=1}:
> d3:={diff(w(x),x)*P3-n3*diff(P3,x)*w(x)=0, w(0)=1}:
> deq:=poltodiffeq(u(x)*v(x)*w(x),[d1,d2,d3],[u(x),v(x),w(x)],y(x)):
> rec:=diffeqtorec(deq,y(x),u(n)); pro:=rectoproc(rec,u(n));
> co3000:=pro(3000); time()-st;
```

$3973942265580043039696 \cdots [1379 \text{ digits}] \cdots 9071342944579342047632 0$

0.24 seconds

Example [Flajolet, Salvy, 1997]

What is the coefficient of x^{3000} in the expansion of

$$(x+1)^{2000}\left(x^2+x+1\right)^{1000}\left(x^4+x^3+x^2+x+1\right)^{500}$$

```
> st:=time(); n1:=2000; n2:=1000; n3:=500;
> P1:=x+1; P2:=x^2+x+1; P3:=x^4+x^3+x^2+x+1;
> d1:={diff(u(x),x)*P1-n1*diff(P1,x)*u(x)=0, u(0)=1}:
> d2:={diff(v(x),x)*P2-n2*diff(P2,x)*v(x)=0, v(0)=1}:
> d3:={diff(w(x),x)*P3-n3*diff(P3,x)*w(x)=0, w(0)=1}:
> deq:=poltodiffeq(u(x)*v(x)*w(x),[d1,d2,d3],[u(x),v(x),w(x)],y(x)):
> rec:=diffeqtorec(deq,y(x),u(n)); pro:=rectoproc(rec,u(n));
> co3000:=pro(3000); time()-st;
```

$3973942265580043039696 \cdots$ [1379 digits] $\cdots 9071342944579342047632 0$

0.24 seconds

```
> st:=time(): co:=coeff(P,x,3000):
> co-co3000, time()-st:
```

0, 93.57 seconds

Baby steps / giant steps

Baby steps / giant steps for polynomial evaluation

Problem: Given a $\mathbb{K}$-algebra $\mathbb{A}$, $a \in \mathbb{A}$ and $P \in \mathbb{K}[x]_{<N}$, compute $P(a)$

Baby steps / giant steps for polynomial evaluation

Problem: Given a $\mathbb{K}$-algebra $\mathbb{A}$, $a \in \mathbb{A}$ and $P \in \mathbb{K}[x]_{<N}$, compute $P(a)$

Horner's rule: $O(N)$ products in $\mathbb{A}$

Baby steps / giant steps for polynomial evaluation

Problem: Given a $\mathbb{K}$-algebra $\mathbb{A}$, $a \in \mathbb{A}$ and $P \in \mathbb{K}[x]_{<N}$, compute $P(a)$

Horner's rule: $O(N)$ products in $\mathbb{A}$

Better algorithm [Paterson-Stockmeyer, 1973]: $O(\sqrt{N})$ products in $\mathbb{A}$
Write $P(x) = P_0(x) + \cdots + P_{\ell-1}(x) \cdot (x^\ell)^{\ell-1}$, with $\ell = \sqrt{N}$ and $\deg(P_i) < \ell$

Baby steps / giant steps for polynomial evaluation

Problem: Given a $\mathbb{K}$-algebra $\mathbb{A}$, $a \in \mathbb{A}$ and $P \in \mathbb{K}[x]_{<N}$, compute $P(a)$

Horner's rule: $O(N)$ products in $\mathbb{A}$

Better algorithm [Paterson-Stockmeyer, 1973]: $O(\sqrt{N})$ products in $\mathbb{A}$

Write $P(x) = P_0(x) + \cdots + P_{\ell-1}(x) \cdot (x^\ell)^{\ell-1}$, with $\ell = \sqrt{N}$ and $\deg(P_i) < \ell$

(BS) Compute $a^2, \quad \ldots, \quad a^\ell =: b$ $O(\sqrt{N})$ products in $\mathbb{A}$

Baby steps / giant steps for polynomial evaluation

Problem: Given a $\mathbb{K}$-algebra $\mathbb{A}$, $a \in \mathbb{A}$ and $P \in \mathbb{K}[x]_{<N}$, compute $P(a)$

Horner's rule: $O(N)$ products in $\mathbb{A}$

Better algorithm [Paterson-Stockmeyer, 1973]: $O(\sqrt{N})$ products in $\mathbb{A}$

Write $P(x) = P_0(x) + \cdots + P_{\ell-1}(x) \cdot (x^\ell)^{\ell-1}$, with $\ell = \sqrt{N}$ and $\deg(P_i) < \ell$

(BS) Compute $a^2, \quad \ldots, \quad a^\ell =: b$ $O(\sqrt{N})$ products in $\mathbb{A}$

(GS) Compute $b_0 = 1,\ b_1 = b, \quad \ldots, \quad b_{\ell-1} = b^{\ell-1}$ $O(\sqrt{N})$ products in $\mathbb{A}$

Baby steps / giant steps for polynomial evaluation

Problem: Given a $\mathbb{K}$-algebra $\mathbb{A}$, $a \in \mathbb{A}$ and $P \in \mathbb{K}[x]_{<N}$, compute $P(a)$

Horner's rule: $O(N)$ products in $\mathbb{A}$

Better algorithm [Paterson-Stockmeyer, 1973]: $O(\sqrt{N})$ products in $\mathbb{A}$

Write $P(x) = P_0(x) + \cdots + P_{\ell-1}(x) \cdot (x^\ell)^{\ell-1}$, with $\ell = \sqrt{N}$ and $\deg(P_i) < \ell$

(BS) Compute $a^2, \quad \ldots, \quad a^\ell =: b$ — $O(\sqrt{N})$ products in $\mathbb{A}$

(GS) Compute $b_0 = 1,\ b_1 = b, \quad \ldots, \quad b_{\ell-1} = b^{\ell-1}$ — $O(\sqrt{N})$ products in $\mathbb{A}$

Evaluate $c_0 = P_0(a), \quad \ldots, \quad c_{\ell-1} = P_{\ell-1}(a)$ — no product in $\mathbb{A}$

Baby steps / giant steps for polynomial evaluation

Problem: Given a $\mathbb{K}$-algebra $\mathbb{A}$, $a \in \mathbb{A}$ and $P \in \mathbb{K}[x]_{<N}$, compute $P(a)$

Horner's rule: $O(N)$ products in $\mathbb{A}$

Better algorithm [Paterson-Stockmeyer, 1973]: $O(\sqrt{N})$ products in $\mathbb{A}$

Write $P(x) = P_0(x) + \cdots + P_{\ell-1}(x) \cdot (x^\ell)^{\ell-1}$, with $\ell = \sqrt{N}$ and $\deg(P_i) < \ell$

(BS) Compute $a^2, \quad \ldots, \quad a^\ell =: b$ $O(\sqrt{N})$ products in $\mathbb{A}$

(GS) Compute $b_0 = 1,\ b_1 = b, \quad \ldots, \quad b_{\ell-1} = b^{\ell-1}$ $O(\sqrt{N})$ products in $\mathbb{A}$

Evaluate $c_0 = P_0(a), \quad \ldots, \quad c_{\ell-1} = P_{\ell-1}(a)$ no product in $\mathbb{A}$

Return $P(a) = b_0 c_0 + \cdots + b_{\ell-1} c_{\ell-1}$ $O(\sqrt{N})$ products in $\mathbb{A}$

Problem: Given a $\mathbb{K}$-algebra $\mathbb{A}$, $a \in \mathbb{A}$ and $P \in \mathbb{K}[x]_{<N}$, compute $P(a)$

Horner's rule: $O(N)$ products in $\mathbb{A}$

Better algorithm [Paterson-Stockmeyer, 1973]: $O(\sqrt{N})$ products in $\mathbb{A}$

Write $P(x) = P_0(x) + \cdots + P_{\ell-1}(x) \cdot (x^\ell)^{\ell-1}$, with $\ell = \sqrt{N}$ and $\deg(P_i) < \ell$

(BS) Compute $a^2, \quad \ldots, \quad a^\ell =: b$ $O(\sqrt{N})$ products in $\mathbb{A}$

(GS) Compute $b_0 = 1,\ b_1 = b, \quad \ldots, \quad b_{\ell-1} = b^{\ell-1}$ $O(\sqrt{N})$ products in $\mathbb{A}$

Evaluate $c_0 = P_0(a), \quad \ldots, \quad c_{\ell-1} = P_{\ell-1}(a)$ no product in $\mathbb{A}$

Return $P(a) = b_0c_0 + \cdots + b_{\ell-1}c_{\ell-1}$ $O(\sqrt{N})$ products in $\mathbb{A}$

Application: evaluation of $P \in \mathbb{K}[x]_{<N}$ at a matrix in $\mathcal{M}_r(\mathbb{K})$ $O(\sqrt{N}\, r^\omega)$

Baby steps / giant steps, application to factorials

Problem: Compute $N! = 1 \times 2 \times \cdots \times N$

Problem: Compute $N! = 1 \times 2 \times \cdots \times N$

Naive algorithm: unroll the recurrence $O(N)$

Baby steps / giant steps, application to factorials

Problem: Compute $N! = 1 \times 2 \times \cdots \times N$

Naive algorithm: unroll the recurrence $O(N)$

Better algorithm [Strassen, 1976]: BS-GS strategy $O\big(\mathsf{M}(\sqrt{N}\,)\log N\big)$

Problem: Compute $N! = 1 \times 2 \times \cdots \times N$

Naive algorithm: unroll the recurrence $O(N)$

Better algorithm [Strassen, 1976]: BS-GS strategy $O\big(\mathsf{M}(\sqrt{N}\,)\log N\big)$

(BS) Compute $P = (x+1)(x+2)\cdots(x+\sqrt{N})$ $O\big(\mathsf{M}(\sqrt{N}\,)\log N\big)$

Baby steps / giant steps, application to factorials

Problem: Compute $N! = 1 \times 2 \times \cdots \times N$

Naive algorithm: unroll the recurrence $O(N)$

Better algorithm [Strassen, 1976]: BS-GS strategy $O\big(\mathsf{M}(\sqrt{N}\,)\log N\big)$

(BS) Compute $P = (x+1)(x+2)\cdots(x+\sqrt{N})$ $O\big(\mathsf{M}(\sqrt{N}\,)\log N\big)$

(GS) Evaluate P at $0, \sqrt{N}, 2\sqrt{N}, \ldots, (\sqrt{N}-1)\sqrt{N}$ $O\big(\mathsf{M}(\sqrt{N}\,)\log N\big)$

Baby steps / giant steps, application to factorials

Problem: Compute $N! = 1 \times 2 \times \cdots \times N$

Naive algorithm: unroll the recurrence $O(N)$

Better algorithm [Strassen, 1976]: BS-GS strategy $O\big(\mathsf{M}(\sqrt{N})\log N\big)$

(BS) Compute $P = (x+1)(x+2)\cdots(x+\sqrt{N})$ $O\big(\mathsf{M}(\sqrt{N})\log N\big)$

(GS) Evaluate P at $0, \sqrt{N}, 2\sqrt{N}, \ldots, (\sqrt{N}-1)\sqrt{N}$ $O\big(\mathsf{M}(\sqrt{N})\log N\big)$

Return $u_N = P((\sqrt{N}-1)\sqrt{N})\cdots P(\sqrt{N})\cdot P(0)$ $O(\sqrt{N})$

Baby steps / giant steps, application to recurrences

Problem: Compute the N-th term u_N of a P-recursive sequence

$$p_r(n)u_{n+r} + \cdots + p_0(n)u_n = 0, \qquad (n \in \mathbb{N})$$

Problem: Compute the N-th term u_N of a P-recursive sequence

$$p_r(n)u_{n+r} + \cdots + p_0(n)u_n = 0, \qquad (n \in \mathbb{N})$$

Naive algorithm: unroll the recurrence $\qquad O(N)$

Baby steps / giant steps, application to recurrences

Problem: Compute the N-th term u_N of a P-recursive sequence

$$p_r(n)u_{n+r} + \cdots + p_0(n)u_n = 0, \qquad (n \in \mathbb{N})$$

Naive algorithm: unroll the recurrence $O(N)$

Better algorithm: $U_n = (u_n, \ldots, u_{n+r-1})^T$ satisfies the 1st order recurrence

$$U_{n+1} = \frac{1}{p_r(n)} A(n) U_n \quad \text{with} \quad A(n) = \begin{bmatrix} & p_r(n) & & \\ & & \ddots & \\ & & & p_r(n) \\ -p_0(n) & -p_1(n) & \ldots & -p_{r-1}(n) \end{bmatrix}.$$

Problem: Compute the N-th term u_N of a P-recursive sequence

$$p_r(n)u_{n+r} + \cdots + p_0(n)u_n = 0, \qquad (n \in \mathbb{N})$$

Naive algorithm: unroll the recurrence $O(N)$

Better algorithm: $U_n = (u_n, \ldots, u_{n+r-1})^T$ satisfies the 1st order recurrence

$$U_{n+1} = \frac{1}{p_r(n)} A(n) U_n \quad \text{with} \quad A(n) = \begin{bmatrix} & p_r(n) & & \\ & & \ddots & \\ & & & p_r(n) \\ -p_0(n) & -p_1(n) & \ldots & -p_{r-1}(n) \end{bmatrix}.$$

$\Longrightarrow u_N$ reads off the matrix factorial $A(N-1) \cdots A(0)$

Problem: Compute the N-th term u_N of a P-recursive sequence

$$p_r(n)u_{n+r} + \cdots + p_0(n)u_n = 0, \qquad (n \in \mathbb{N})$$

Naive algorithm: unroll the recurrence $O(N)$

Better algorithm: $U_n = (u_n, \ldots, u_{n+r-1})^T$ satisfies the 1st order recurrence

$$U_{n+1} = \frac{1}{p_r(n)} A(n) U_n \quad \text{with} \quad A(n) = \begin{bmatrix} & p_r(n) & & \\ & & \ddots & \\ & & & p_r(n) \\ -p_0(n) & -p_1(n) & \ldots & -p_{r-1}(n) \end{bmatrix}.$$

$\Longrightarrow u_N$ reads off the matrix factorial $A(N-1)\cdots A(0)$

[Chudnovsky-Chudnovsky, 1987]: (BS)-(GS) strategy $O\left(\mathsf{M}(\sqrt{N}\,)\log N\right)$

Baby steps / giant steps, application to point counting

Problem: count the number n of solutions of the equation $y^2 = f(x)$ over $\mathbb{F}_p$

Baby steps / giant steps, application to point counting

Problem: count the number n of solutions of the equation $y^2 = f(x)$ over $\mathbb{F}_p$

Basic idea [Deuring, 1941]: if $\deg(f) = 3$, then $n \bmod p = -\,[x^{p-1}]f(x)^{\frac{p-1}{2}}$

Baby steps / giant steps, application to point counting

Problem: count the number n of solutions of the equation $y^2 = f(x)$ over $\mathbb{F}_p$

Basic idea [Deuring, 1941]: if $\deg(f) = 3$, then $n \bmod p = -[x^{p-1}]f(x)^{\frac{p-1}{2}}$

Explanation: z is a non-zero square in $\mathbb{F}_p$ exactly when $z^{(p-1)/2} = 1$

Baby steps / giant steps, application to point counting

Problem: count the number n of solutions of the equation $y^2 = f(x)$ over $\mathbb{F}_p$

Basic idea [Deuring, 1941]: if $\deg(f) = 3$, then $n \bmod p = -[x^{p-1}]f(x)^{\frac{p-1}{2}}$

Explanation: z is a non-zero square in $\mathbb{F}_p$ exactly when $z^{(p-1)/2} = 1$

Generalization [Cartier-Manin, 1956]: if $\deg(f) = 2g + 1$, then $n \bmod p$ reads off the Hasse-Witt matrix $(h_{i,j})_{i,j=1}^{g}$, with $h_{i,j} = [x^{ip-j}]f(x)^{(p-1)/2}$

Problem: count the number n of solutions of the equation $y^2 = f(x)$ over $\mathbb{F}_p$

Basic idea [Deuring, 1941]: if $\deg(f) = 3$, then $n \bmod p = -[x^{p-1}]f(x)^{\frac{p-1}{2}}$

Explanation: z is a non-zero square in $\mathbb{F}_p$ exactly when $z^{(p-1)/2} = 1$

Generalization [Cartier-Manin, 1956]: if $\deg(f) = 2g+1$, then $n \bmod p$ reads off the Hasse-Witt matrix $(h_{i,j})_{i,j=1}^{g}$, with $h_{i,j} = [x^{ip-j}]f(x)^{(p-1)/2}$

Corollary [B-Gaudry-Schost, 2007]: $\tilde{O}(\sqrt{p})$ hyperelliptic point counting / $\mathbb{F}_p$

Problem: count the number n of solutions of the equation $y^2 = f(x)$ over $\mathbb{F}_p$

Basic idea [Deuring, 1941]: if $\deg(f) = 3$, then $n \bmod p = -\,[x^{p-1}]f(x)^{\frac{p-1}{2}}$

Explanation: z is a non-zero square in $\mathbb{F}_p$ exactly when $z^{(p-1)/2} = 1$

Generalization [Cartier-Manin, 1956]: if $\deg(f) = 2g+1$, then $n \bmod p$ reads off the Hasse-Witt matrix $(h_{i,j})_{i,j=1}^{g}$, with $h_{i,j} = [x^{ip-j}]f(x)^{(p-1)/2}$

Corollary [B-Gaudry-Schost, 2007]: $\tilde{O}(\sqrt{p})$ hyperelliptic point counting / $\mathbb{F}_p$

▷ Based on [Flajolet-Salvy, 1997]: $h = f^N$ satisfies the differential equation $fh' - Nf'h = 0$, thus its coefficient sequence is P-recursive.

(1) Show that if $P \in \mathbb{K}[x]$ has degree d, then the sequence $(P(n))_{n\geq 0}$ is C-recursive, and admits $(x-1)^{d+1}$ as a characteristic polynomial.

(2) Let $P = \sum_{i=0}^{2N} p_i x^i \in \mathbb{Z}[X]$ be the polynomial $P(x) = (1+x+x^2)^N$.

1. Show that the parity of all coefficients of P can be determined in $O(\mathsf{M}(N))$ bit ops.
2. Show that P satisfies a linear differential equation of order 1 with polynomial coefficients.
3. Determine a linear recurrence of order 2 satisfied by the sequence $(p_i)_i$.
4. Give an algorithm that computes p_N in $\tilde{O}(N)$ bit ops.