N-th term of a linear recurrent sequence

Alin Bostan

Inria — informatics mathematics

MPRI

C-2-22

October 25, 2022

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.

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 $\tilde{O}(N^2)$ 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: $U_n = (u_n, \dots, u_{n+r-1})^T$ satisfies the 1st order recurrence

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

$\Longrightarrow u_N$ 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));
```

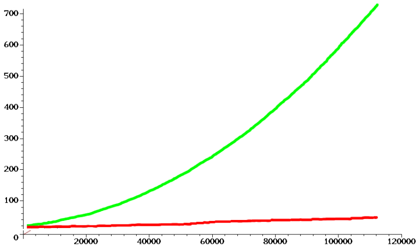
```
pro := proc(n::nonnegint)
local i1, loc0, loc1, loc2, tmp2, tmp1, i2;
  if n <= 22 then
    loc0 := 1;  loc1 := 2;
    if n = 0 then return loc0
      else for i1 to n - 1 do
         loc2 := (-loc0 + loc1 + loc1*(i1 + 1))/(i1 + 1);
         loc0 := loc1; loc1 := loc2
      end do
    end if; loc1
  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 ⟶ 4 billion digits [Chudnovsky-Chudnovsky 1994]

▷ Current record: 100 trillion digits [Iwao 2022]

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(); n3:=500; n1:=4*n3; n2:=2*n3;
> 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(); n3:=500; n1:=4*n3; n2:=2*n3;
> 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;
```

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

0.24 seconds

```
> st:=time(): P:=expand(P1^n1*P2^n2*P3^n3): 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 $\deg(P_i) < \ell$

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

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

Evaluate $c_0 = P_0(a), \quad \dots, \quad c_{\ell-1} = P_{\ell-1}(a)$ 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 $P(a) = b_0 c_0 + \cdots + b_{\ell-1} c_{\ell-1}$ — $O(\sqrt{N})$ products in $\mathbb{A}$

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

Baby steps / giant steps, application to factorials

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

Baby steps / giant steps, application to factorials

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

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

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

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

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

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 $O(N)$

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 $O\big(\mathsf{M}(\sqrt{N}\,)\log N\big)$

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 $n \bmod p = -[x^{p-1}]f(x)^{\frac{p-1}{2}}$

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 $z^{(p-1)/2} = 1$

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.

Two exercises for next time (8/11/2022)

(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.

Deduce that P can be evaluated at the $N \gg d$ points $1, 2, \ldots, N$ in $O(N\,\mathsf{M}(d)/d)$ operations in $\mathbb{K}$.

(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.