Algebraic power series mod p
—fast computation of coefficients—

Alin Bostan

Inria — informatics mathematics

♦ SLC'80 ♦ Lyon ♦ March 28, 2018 ♦

One of the most difficult questions in modular computations is the complexity of computations mod p for a large prime p of coefficients in the expansion of an algebraic function.

[D. Chudnovsky & G. Chudnovsky, 1990]
Computer Algebra in the Service of Mathematical Physics and Number Theory

Main objects and goal

- p, a prime number
- N, a positive integer
- $\mathbb{F}_p$, the finite field with p elements
- $a \in \mathbb{F}_p$
- $E(t,y) \in \mathbb{F}_p[t,y]$, irreducible with $E(0,a) = 0$ and $\frac{\partial E}{\partial y}(0,a) \neq 0$
- $f(t) = \sum_n f_n t^n$, the unique root in $\mathbb{F}_p[[t]]$ of $E(t,f(t)) = 0$ with $f(0) = a$

Main objects and goal

- p, a prime number
- N, a positive integer
- $\mathbb{F}_p$, the finite field with p elements
- $a \in \mathbb{F}_p$
- $E(t,y) \in \mathbb{F}_p[t,y]$, irreducible with $E(0,a) = 0$ and $\frac{\partial E}{\partial y}(0,a) \neq 0$
- $f(t) = \sum_n f_n t^n$, the unique root in $\mathbb{F}_p[[t]]$ of $E(t,f(t)) = 0$ with $f(0) = a$

Goal: design efficient algorithms for computing f_N

Main objects and goal

- p, a prime number
- N, a positive integer
- $\mathbb{F}_p$, the finite field with p elements
- $a \in \mathbb{F}_p$
- $E(t,y) \in \mathbb{F}_p[t,y]$, irreducible with $E(0,a) = 0$ and $\frac{\partial E}{\partial y}(0,a) \neq 0$
- $f(t) = \sum_n f_n t^n$, the unique root in $\mathbb{F}_p[[t]]$ of $E(t, f(t)) = 0$ with $f(0) = a$

Goal: design efficient algorithms for computing f_N

▷ Efficiency: measured in terms of bit operations (Turing machine model)

▷ Assume: input E has degree $d = \deg_y E$ and height $h = \deg_t E$, both $O(1)$

A special case: $d = 1$

- $f(t) = \sum_n f_n t^n \in \mathbb{F}_p[[t]] \cap \mathbb{F}_p(t) \iff (f_n)_n$ satisfies a recurrence

$$f_{n+h} = c_{h-1} f_{n+h-1} + \cdots + c_0 f_n, \qquad n \geq 0.$$

- This recurrence rewrites in matrix form

$$\underbrace{\begin{bmatrix} f_N \\ f_{N+1} \\ \vdots \\ f_{N+h-1} \end{bmatrix}}_{V_N} = \underbrace{\begin{bmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ c_0 & c_1 & \cdots & c_{h-1} \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} f_{N-1} \\ f_N \\ \vdots \\ f_{N+h-2} \end{bmatrix}}_{V_{N-1}} = C^N \underbrace{\begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_{h-1} \end{bmatrix}}_{V_0}, \qquad N \geq 1.$$

- 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: $O(\log N)$ ops. in $\mathbb{F}_p$, thus $\tilde{O}(\log N \cdot \log p)$ bit ops.

▷ This is an ideal complexity result!

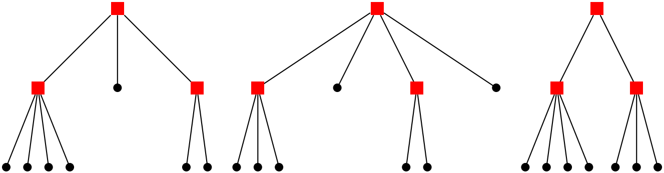
▷ Open question: $\exists$? algorithm of complexity $\mathsf{Poly}(\log N, \log p)$ for $d \geq 2$?

▷ Concrete challenge: for $f = \frac{1}{\sqrt{1-4t}}$, compute $f_N \bmod p = \binom{2N}{N} \bmod p$.

Ten methods to compute f_N

Guiding example

Problem: count 2-3-4 trees $\longrightarrow$ f_n = nb. of trees with n internal nodes ■



▷ Generating function:

$$f = \sum_n f_n t^n = 1 + 3t + 27t^2 + 333t^3 + 4752t^4 + 73764t^5 + \cdots,$$

root of

$$E(t, y) = y - 1 - t(y^2 + y^3 + y^4).$$

$(\log p)$-algorithms

Method 1: non-linear recurrences

- Starting point: the sequence $(f_n)_{n\geq 0}$ satisfies a *non-linear recurrence*.

- This yields an algorithm for f_N using $\mathsf{Poly}(N^d)$ operations in $\mathbb{F}_p$, thus of bit complexity $\tilde{O}(\log p \cdot \mathsf{Poly}(N^d))$.

- E.g., for 2-3-4 trees, with

$$E(t,y) = y - 1 - t(y^2 + y^3 + y^4) \in \mathbb{F}_p[t,y],$$

we have for $n \geq 1$:

$$f_n = \sum_{i+j=n-1} f_i f_j + \sum_{i+j+k=n-1} f_i f_j f_k + \sum_{i+j+k+\ell=n-1} f_i f_j f_k f_\ell.$$

▷ f_N can be computed in $O(N^4)$ ops. in $\mathbb{F}_p$, i.e., in $\tilde{O}(N^4 \cdot \log p)$ bit ops.

Method 2: fixed-point theorem

- Starting point: f is the limit of the sequence $(F_k)_k$ of power series in $\mathbb{F}_p[[t]]$ defined by $F_0 = a$, and $F_{k+1}(t) = F_k(t) - E(t, F_k(t))$ for $k \geq 0$.

- This yields an algorithm for f_N using N products of power series mod t^N

- Each such product can be performed in $\tilde{O}(N)$ ops. in $\mathbb{F}_p$ via FFT

- E.g., for 2-3-4 trees, with

$$E(t,y) = y - 1 - t(y^2 + y^3 + y^4) \in \mathbb{F}_p[t,y],$$

compute:

$$F_0 = 1, \qquad F_{k+1} = 1 + tF_k^2 + tF_k^3 + tF_k^4 \mod t^{N+1} \quad \text{for} \quad 0 \leq k \leq N.$$

▷ f_N can be computed in $\tilde{O}(N^2)$ ops. in $\mathbb{F}_p$, i.e., in $\tilde{O}(N^2 \cdot \log p)$ bit ops.

Method 3: Newton iteration

• Starting point: f is the limit of the sequence $(G_k)_k$ of power series in $\mathbb{F}_p[[t]]$ defined by $G_0 = a$, and $G_{k+1}(t) = G_k(t) - \dfrac{E(t, G_k(t))}{\frac{\partial E}{\partial y}(t, G_k(t))}$ for $k \geq 0$.

• This yields an algorithm for f_N using products of power series mod t^{2^k}, for $k = 1, \ldots, \log N$

• Each such product can be done in $\tilde{O}(2^k)$ ops. in $\mathbb{F}_p$, for a total of $\tilde{O}(N)$

• E.g., for 2-3-4 trees, with

$$E(t, y) = y - 1 - t(y^2 + y^3 + y^4) \in \mathbb{F}_p[t, y],$$

compute:

$$G_0 = 1, \qquad G_{k+1} = G_k - \frac{G_k - (1 + tG_k^2 + tG_k^3 + tG_k^4)}{1 - 2tG_k - 3tG_k^2 - 4tG_k^3} \mod t^{2^{k+1}} \quad \text{for} \quad k \geq 0.$$

▷ f_N can be computed in $\tilde{O}(N)$ ops. in $\mathbb{F}_p$, i.e., in $\tilde{O}(N \cdot \log p)$ bit ops.
[Lipson, 1976; Kung, Traub, 1978]

Method 4: linear recurrences

- Starting point:

Abel's theorem (1827)

The sequence $(f_i)_i$ satisfies a linear recurrence with *polynomial coefficients*

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

- This yields an algorithm for f_N using $O(N)$ operations in $\mathbb{F}_p$ $^{(\dagger)}$
- E.g., for 2-3-4 trees, with

$$E(t,y) = y - 1 - t(y^2 + y^3 + y^4) \in \mathbb{F}_p[t,y],$$

determine, then unroll, the recurrence:

$$162\,n\,(n+1)\,(2\,n+1)\,f_n + 108\,(n+1)\left(26\,n^2 + 77\,n + 63\right) f_{n+1} + \\ \cdots + 75\,(3\,n+17)\,(3\,n+19)\,(n+6)\,f_{n+6} = 0.$$

▷ f_N can be computed in $O(N)$ ops. in $\mathbb{F}_p$, i.e., in $\tilde{O}(N \cdot \log p)$ bit ops.$^{(\dagger)}$

$^{(\dagger)}$ Under the additional assumption that $p_r(n) \neq 0$ for $n = 0, \ldots, N$.

Method 5: linear recurrences, and baby-steps / giant-steps

- Starting point: Abel's theorem, combined with the following strategy
- $U_n = (f_n, \ldots, f_{n+r-1})^T$ satisfies the 1st order matrix recurrence

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

$\Longrightarrow f_N$ reads off the matrix factorial $A(N-1) \cdots A(0)^{(\dagger)}$

- [Chudnovsky-Chudnovsky, 1987]: (BS)–(GS) strategy

(BS) Compute $P = A(x + \sqrt{N} - 1) \cdots A(x+1)A(x)$ $\quad \tilde{O}(\sqrt{N})$

(GS) Evaluate P at $0, \sqrt{N}, 2\sqrt{N}, \ldots, (\sqrt{N}-1)\sqrt{N}$ $\quad \tilde{O}(\sqrt{N})$

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

▷ f_N can be computed in $\tilde{O}(\sqrt{N})$ ops. in $\mathbb{F}_p$, i.e., in $\tilde{O}(\sqrt{N} \cdot \log p)$ bit ops.$^{(\dagger)}$

$^{(\dagger)}$ Under the additional assumption that $p_r(n) \neq 0$ for $n = 0, \ldots, N$.

Recap: $(\log p)$-algorithms

Method	arithmetic complexity	bit complexity
1. non-linear recurrences	$\tilde{O}(N^d)$	$\tilde{O}(N^d \cdot \log p)$
2. fixed-point theorem	$\tilde{O}(N^2)$	$\tilde{O}(N^2 \cdot \log p)$
3. Newton iteration	$\tilde{O}(N)$	$\tilde{O}(N \cdot \log p)$
4. linear recurrences	$O(N)$	$\tilde{O}(N \cdot \log p)$
5. baby-steps / giant-steps	$\tilde{O}(\sqrt{N})$	$\tilde{O}(\sqrt{N} \cdot \log p)$

$(\log N)$-algorithms

General strategy

• Starting point:

Christol's theorem (1979)

If $f \in \mathbb{F}_p[[t]]$ is algebraic, then there exists an $\mathbb{F}_p$-vector space V such that:

- $\dim_{\mathbb{F}_p} V < +\infty$,
- V contains f,
- V is left stable by the section (Cartier) operators S_r $(0 \leq r < p)$

$$S_r\left(c_0 + c_1 t + c_2 t^2 + \cdots\right) = c_r + c_{r+p} t + c_{r+2p} t^2 + \cdots .$$

• Each choice of V yields an algorithm for f_N using $\log_p N$ applications of the section operators on elements in V:

$$\text{if} \quad N = \overline{(r_\ell \cdots r_1 r_0)}_p \quad \text{then} \quad f_N = \left(S_{r_\ell} \cdots S_{r_1} S_{r_0} f\right)(0)$$

▷ The choice of V has an impact on the complexity!

▷ Different proofs of Christol's theorem lead to different $(\log N)$-algorithms.

Method 6: Mahler equations

- Starting point: If $f \in \mathbb{F}_p[[t]]$ is algebraic, then it satisfies a Mahler equation

$$a_0(t)f(t) + a_1(t)f(t^p) + \cdots + a_r(t)f(t^{p^r}) = 0,$$

with coefficients a_j in $\mathbb{F}_p[t]$ with $\deg(a_j) \leq \mathsf{Poly}(p^d)$, and $a_0 \neq 0$.

- Then for some $N \leq \mathsf{Poly}(p^d)$, one may take in Christol's theorem

$$V = \mathrm{Vect}_{\mathbb{F}_p} \left\langle \sum_{i=0}^{r} c_i (f/a_0)^{p^i},\ c_i \in \mathbb{F}_p[t]_{\leq N} \right\rangle$$

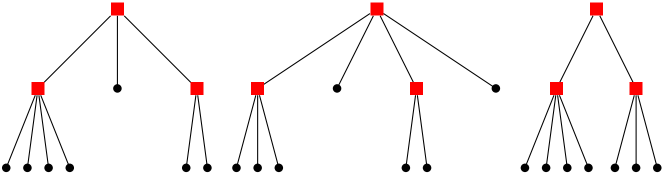
[Christol, Kamae, Mendès France, Rauzy, 1980]

- This yields an algorithm for f_N using $O(\log N)$ sections in V.

▷ f_N can be computed using $\tilde{O}(\mathsf{Poly}(p^d) \cdot \log N)$ bit ops.

Example: counting 2-3-4 trees modulo 2

Problem: count 2-3-4 trees $\longrightarrow$ f_n = nb. of trees with n internal nodes ■



$f = \sum_n f_n t^n = 1 + 3t + 27t^2 + 333t^3 + \cdots$, root of $E = y - 1 - t(y^2 + y^3 + y^4)$

▷ Mahler equation $t + f(t) + (t^2 + t + 1)f(t^2) + tf(t^4) + t^2 f(t^8) = 0 \mod 2$

▷ $f_n \bmod 2 = \begin{cases} f_{(n-1)/2} \bmod 2, & \text{if } n \equiv 3 \bmod 4. \\ f_{(n-1)/2} + f_{(n-1)/4} \bmod 2, & \text{if } n \equiv 1 \bmod 4, \\ f_{n/2} + f_{n/2-1} + f_{(n-2)/8} \bmod 2, & \text{if } n \equiv 2 \bmod 8, \\ f_{n/2} + f_{n/2-1} \bmod 2, & \text{else.} \end{cases}$

▷ Computation of f_N modulo 2 in $O(\log N)$ bit operations.

Method 7: diagonals

- Starting point:

Furstenberg's theorem (1967)

If $E(0,0) = 0$ and $\frac{\partial E}{\partial y}(0,0) \neq 0$, then

$$f(t) = \operatorname{Diag}(g) \quad \text{where} \quad g(x,y) = \frac{y \cdot \frac{\partial E}{\partial y}(xy,y)}{y^{-1} \cdot E(xy,y)}.$$

- Christol's argument (1979): Since $f(t) = \operatorname{Diag} \dfrac{a(x,y)}{b(x,y)}$, we have

$S_r f(t)$

- Starting point:

Furstenberg's theorem (1967)

If $E(0,0) = 0$ and $\frac{\partial E}{\partial y}(0,0) \neq 0$, then

$$f(t) = \mathrm{Diag}\,(g) \quad \text{where} \quad g(x,y) = \frac{y \cdot \frac{\partial E}{\partial y}(xy,y)}{y^{-1} \cdot E(xy,y)}.$$

- Christol's argument (1979): Since $f(t) = \mathrm{Diag}\,\dfrac{a(x,y)}{b(x,y)}$, we have

$$S_r f(t) = S_r\left(\mathrm{Diag}\,\frac{a(x,y)}{b(x,y)}\right)$$

Method 7: diagonals

- Starting point:

Furstenberg's theorem (1967)

If $E(0,0) = 0$ and $\frac{\partial E}{\partial y}(0,0) \neq 0$, then

$$f(t) = \mathrm{Diag}\,(g) \quad \text{where} \quad g(x,y) = \frac{y \cdot \frac{\partial E}{\partial y}(xy,y)}{y^{-1} \cdot E(xy,y)}.$$

- Christol's argument (1979): Since $f(t) = \mathrm{Diag}\,\dfrac{a(x,y)}{b(x,y)}$, we have

$$S_r f(t) = S_r\left(\mathrm{Diag}\,\frac{a(x,y)}{b(x,y)}\right) = \mathrm{Diag}\,S_r\left(\frac{a(x,y)}{b(x,y)}\right)$$

Method 7: diagonals

- Starting point:

Furstenberg's theorem (1967)

If $E(0,0)=0$ and $\frac{\partial E}{\partial y}(0,0)\neq 0$, then
$$f(t)=\mathrm{Diag}\,(g)\quad \text{where}\quad g(x,y)=\frac{y\cdot\frac{\partial E}{\partial y}(xy,y)}{y^{-1}\cdot E(xy,y)}.$$

- Christol's argument (1979): Since $f(t)=\mathrm{Diag}\,\dfrac{a(x,y)}{b(x,y)}$, we have

$$S_r f(t) = S_r\left(\mathrm{Diag}\,\frac{a(x,y)}{b(x,y)}\right)=\mathrm{Diag}\ S_r\left(\frac{a(x,y)}{b(x,y)}\right)=\mathrm{Diag}\ \frac{S_r\left(a(x,y)b(x,y)^{p-1}\right)}{b(x,y)},$$

Method 7: diagonals

- Starting point:

Furstenberg's theorem (1967)

If $E(0,0) = 0$ and $\frac{\partial E}{\partial y}(0,0) \neq 0$, then

$$f(t) = \mathrm{Diag}\,(g) \quad \text{where} \quad g(x,y) = \frac{y \cdot \frac{\partial E}{\partial y}(xy,y)}{y^{-1} \cdot E(xy,y)}.$$

- Christol's argument (1979): Since $f(t) = \mathrm{Diag}\, \dfrac{a(x,y)}{b(x,y)}$, we have

$$S_r f(t) = S_r\left(\mathrm{Diag}\, \frac{a(x,y)}{b(x,y)}\right) = \mathrm{Diag}\, S_r\left(\frac{a(x,y)}{b(x,y)}\right) = \mathrm{Diag}\, \frac{S_r\left(a(x,y)b(x,y)^{p-1}\right)}{b(x,y)},$$

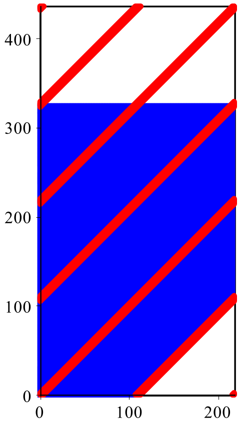
so one may take in Christol's thm $V = \mathrm{Diag}\left(\frac{1}{b(x,y)} \cdot \mathbb{F}_p[x,y]_{\leq(\deg_x b, \deg_y b)}\right)$

- This yields an algorithm for f_N using $O(\log N)$ sections in V.

▷ f_N can be computed using $\tilde{O}(p^2 \cdot \log N)$ bit ops.

Method 8: Partial diagonals

- Starting point: Only a small part of $b(x,y)^{p-1}$ is enough to compute a section in method 7: $O(1)$ strips of width $O(1)$ and length $O(p)$
- Example: $p = 109$, $E = (1+t)(t-y) + t^2y^2 + (1+t)y^3 + y^4$



$\triangleright$ f_N can be computed using $\tilde{O}(p \cdot \log N)$ bit ops. [B., Christol, Dumas, 2016]

Method 9: Furstenberg-like and Hermite-Padé approximation

- Starting point:

Theorem [B., Caruso, Christol, Dumas, 2018]

One may take in Christol's theorem

$$V = \left\{ \frac{P(t,f(t))}{\frac{\partial E}{\partial y}(t,f(t))} \quad \middle| \quad P \in \mathbb{F}_p[t,y], \quad \deg_y P < d \text{ and } \deg_t P < h \right\}.$$

- This yields an algorithm for f_N using $O(\log N)$ sections in V.

▷ Applying a section to $g = \frac{P(t,f(t))}{\frac{\partial E}{\partial y}(t,f(t))}$ amounts to:

- expanding $g \bmod t^{2dh\,p}$ $\tilde{O}(p)$ by Newton iterations
- solving a Hermite-Padé approximation problem at order $2dh = O(1)$.

▷ f_N can be computed using $\tilde{O}(p \cdot \log N)$ bit ops.

Method 10: Hermite-Padé and baby-steps / giant-steps

- Starting point: Only $O(1)$ coefficients (of index from p to p) of $g = \frac{P(t,f)}{\frac{\partial E}{\partial y}(t,f)}$ are needed in Method 9! (since we are only interested in one of its sections)

- Idea: g is algebraic, hence D-finite, so use baby-steps / giant-steps to compute those coefficients in $\tilde{O}(\sqrt{p})$ ops.

- Main difficulty to overcome: divisions by p (as in Method 5)

- Solution: lift to p-adics, control precision loss

▷ f_N can be computed using $\tilde{O}(\sqrt{p} \cdot \log N)$ bit ops.

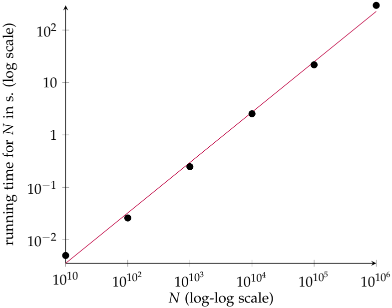
[B., Caruso, Christol, Dumas, 2018]

Overview: Ten methods to compute $f_N \bmod p$

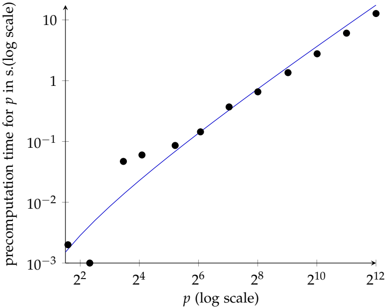
Method	bit complexity
1. non-linear recurrences	$\mathsf{Poly}(N^d \cdot \log p)$
2. fixed-point theorem	$\tilde{O}(N^2 \cdot \log p)$
3. Newton iteration	$\tilde{O}(N \cdot \log p)$
4. linear recurrences	$\tilde{O}(N \cdot \log p)$
5. baby-steps / giant-steps	$\tilde{O}(\sqrt{N} \cdot \log p)$
6. Mahler equations	$\mathsf{Poly}(p^d \cdot \log N)$
7. diagonals	$\tilde{O}(p^2 \cdot \log N)$
8. partial diagonals	$\tilde{O}(p \cdot \log N)$
9. Furstenberg + Hermite-Padé	$\tilde{O}(p \cdot \log N)$
10. Hermite-Padé + BS–GS	$\tilde{O}(\sqrt{p} \cdot \log N)$

Bonus

Timings (method 8)



$O(\log N)$



$\tilde{O}(p)$

A difficult but nice case

For

$$f = \frac{1}{\sqrt{(1-at)^2 - 4t^2}}$$

and

$$N = \frac{p-1}{2},$$

computing $f_N \bmod p$ reduces to computing the number of points $(x, y) \in \mathbb{F}_p \times \mathbb{F}_p$ on the elliptic curve

$$y^2 = x(1 + ax + x^2).$$

This can be done in polynomial time in $\log p$ by Schoof's algorithm (1985).

An amusing conjecture

Let

$$C_n = \frac{1}{n+1}\binom{2n}{n}$$

be the nth Catalan number. Then:

- The last digit (in base 10) of C_n is never 3;
- For $n \gg 0$, the last digit of any odd C_n is always 5.

Thanks for your attention!

Fast multiplication and division of power series

[Schönhage-Strassen, 1971]: FFT-multiplication in $\mathbb{F}_p[x]_{<N}$ using $\tilde{O}(N)$ ops.

[Sieveking-Kung, 1972]: Newton iteration for the reciprocal of $f \in \mathbb{F}_p[[x]]$:

$$g_0 = \frac{1}{f_0} \quad \text{and} \quad g_{\kappa+1} = g_\kappa + g_\kappa(1 - f g_\kappa) \mod x^{2^{\kappa+1}} \quad \text{for } \kappa \geq 0$$

$$\mathsf{R}(N) = \mathsf{R}(N/2) + \tilde{O}(N) \implies \mathsf{R}(N) = \tilde{O}(N)$$

Corollary: Division of power series at precision N in $\tilde{O}(N)$

Given $F, G \in \mathbb{F}_p[x]_{\leq N}$, compute (Q, R) in Euclidean division $F = QG + R$

Schoolbook algorithm: $O(N^2)$

Better idea: look at $F = QG + R$ from the infinity: $Q \sim_{+\infty} F/G$

Formally: Let $N = \deg(F)$, $n = \deg(G)$, then $\deg(Q) = N - n$, $\deg(R) < n$ and

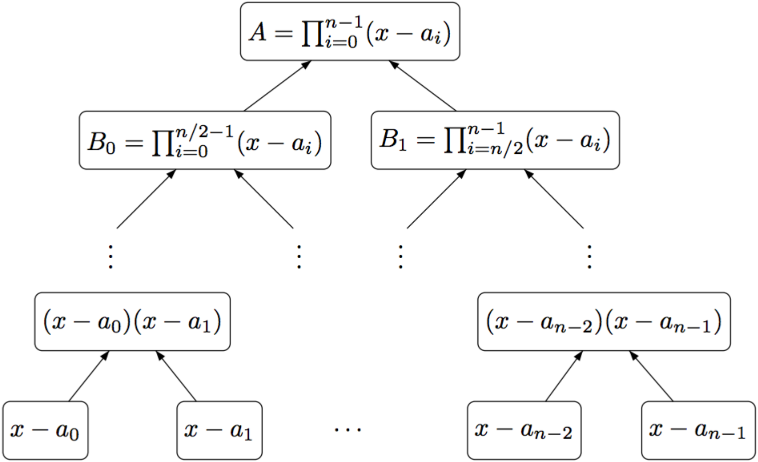
$$\underbrace{F(1/x)x^N}_{\mathrm{rev}(F)} = \underbrace{G(1/x)x^n}_{\mathrm{rev}(G)} \cdot \underbrace{Q(1/x)x^{N-n}}_{\mathrm{rev}(Q)} + \underbrace{R(1/x)x^{\deg(R)}}_{\mathrm{rev}(R)} \cdot x^{N-\deg(R)}$$

Strassen's algorithm [1973]: $\tilde{O}(N)$

- Compute $\mathrm{rev}(Q) = \mathrm{rev}(F)/\mathrm{rev}(G) \mod x^{N-n+1}$ $\tilde{O}(N)$
- Recover Q $O(N)$
- Deduce $R = F - QG$ $\tilde{O}(N)$

Subproduct tree

Problem: Given $a_0, \ldots, a_{n-1} \in \mathbb{F}_p$, compute $A = \prod_{i=0}^{n-1}(x - a_i)$



Cost: $\mathsf{S}(n) = 2 \cdot \mathsf{S}(n/2) + \tilde{O}(n) \implies \mathsf{S}(n) = \tilde{O}(n)$.

Fast multipoint evaluation [Borodin-Moenck, 1974]

$\boxed{\text{Given } a_0, \ldots, a_{n-1} \in \mathbb{F}_p,\ P \in \mathbb{F}_p[x]_{<n}, \text{ compute } P(a_0), \ldots, P(a_{n-1})}$

Naive algorithm: Compute the $P(a_i)$ independently $O(n^2)$

Idea: Use recursively Bézout's identity $P(a) = P(x) \bmod (x - a)$

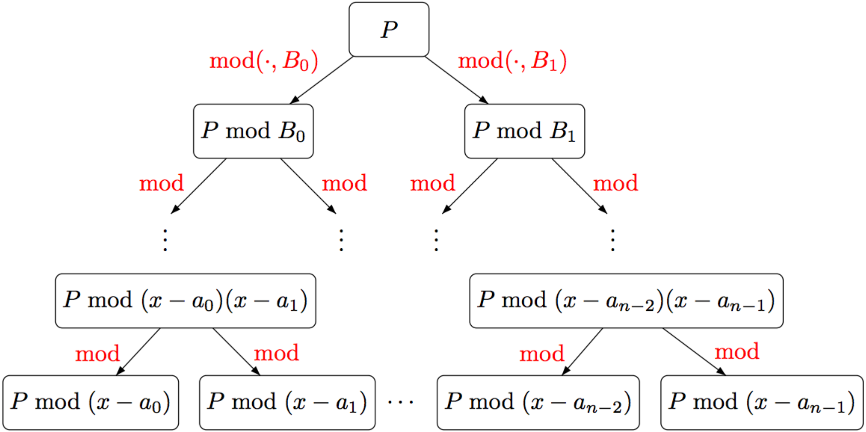
Divide and conquer: FFT-type idea, evaluation by repeated division

- $P_0 = P \mod (x - a_0) \cdots (x - a_{n/2-1})$
- $P_1 = P \mod (x - a_{n/2}) \cdots (x - a_{n-1})$

$$\Longrightarrow \quad \begin{cases} P_0(a_0) = P(a_0), & \ldots, \quad P_0(a_{n/2-1}) = P(a_{n/2-1}) \\ P_1(a_{n/2}) = P(a_{n/2}), & \ldots, \quad P_1(a_{n-1}) = P(a_{n-1}) \end{cases}$$

Fast multipoint evaluation [Borodin-Moenck, 1974]

$\boxed{\text{Given } a_0, \ldots, a_{n-1} \in \mathbb{F}_p,\ P \in \mathbb{F}_p[x]_{<n}, \text{ compute } P(a_0), \ldots, P(a_{n-1})}$



Cost: $\mathsf{E}(n) = 2 \cdot \mathsf{E}(n/2) + \tilde{O}(n) \implies \mathsf{E}(n) = \tilde{O}(n).$