Fast algorithms for polynomials and matrices
(A brief introduction to Computer Algebra)
— Part 2 —

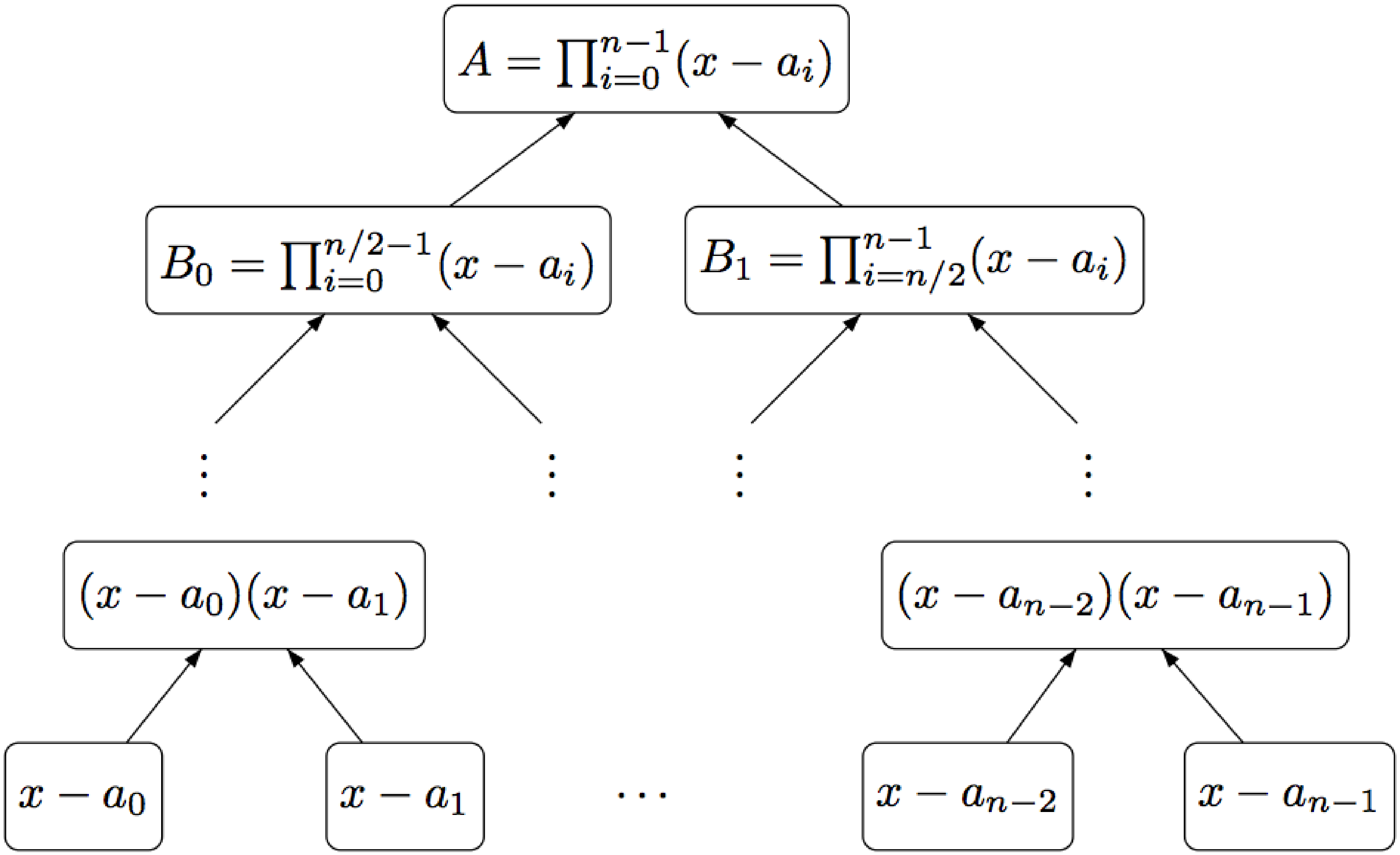
Alin Bostan

SpecFun, INRIA

Seminar on Algebraic Systems Theory
April 25, 2013

Subproduct tree

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



Master Theorem: $\mathsf{C}(n) = 2 \cdot \mathsf{C}(n/2) + O(\mathsf{M}(n)) \implies \mathsf{C}(n) = O(\mathsf{M}(n) \log n)$

Fast multipoint evaluation

[Borodin-Moenck, 1974]

Pb: Given $a_0, \dots, a_{n-1} \in \mathbb{K}$ and $P \in \mathbb{K}[x]_{<n}$, compute $P(a_0), \dots, P(a_{n-1})$

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

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

Divide and conquer: Same idea as for DFT = evaluation by repeated division

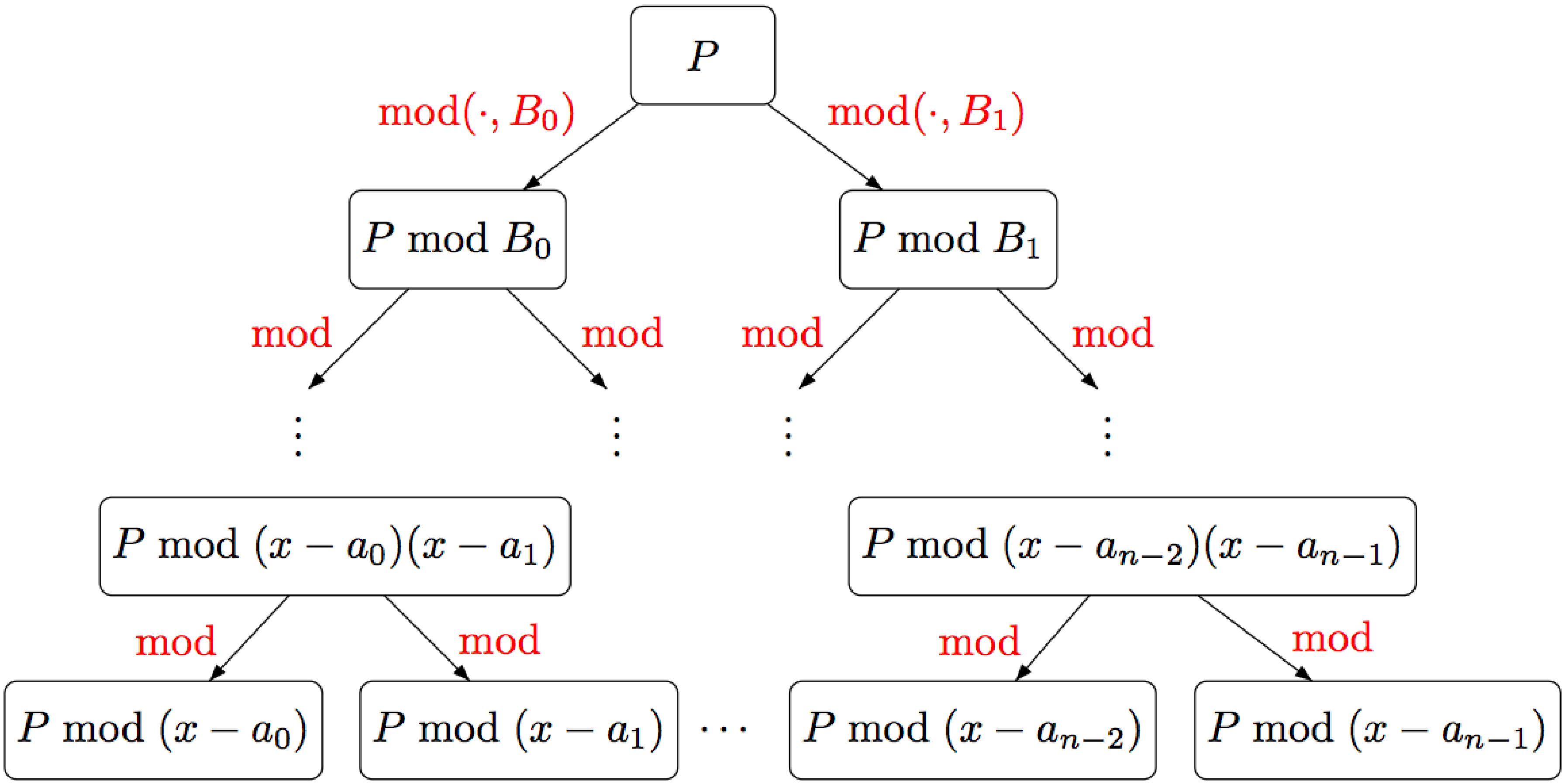
- $P_0 = P \mod (x - a_0) \cdots (x - a_{n/2-1})$
- $P_1 = P \mod (x - a_{n/2}) \cdots (x - a_{n-1})$

$$\Longrightarrow \quad \left\{ \begin{array}{ll} P_0(a_0) = P(a_0), \quad \dots, \quad P_0(a_{n/2-1}) = P(a_{n/2-1}) \\ P_1(a_{n/2}) = P(a_{n/2}), \quad \dots, \quad P_1(a_{n-1}) = P(a_{n-1}) \end{array} \right.$$

Fast multipoint evaluation

[Borodin-Moenck, 1974]

Pb: Given $a_0, \ldots, a_{n-1} \in \mathbb{K}$ and $P \in \mathbb{K}[x]_{<n}$, compute $P(a_0), \ldots, P(a_{n-1})$



Master Theorem: $\mathsf{C}(n) = 2 \cdot \mathsf{C}(n/2) + O(\mathsf{M}(n)) \implies \mathsf{C}(n) = O(\mathsf{M}(n) \log n)$

Fast interpolation

[Borodin-Moenck, 1974]

Problem: Given $a_0, \ldots, a_{n-1} \in \mathbb{K}$ mutually distinct, and $v_0, \ldots, v_{n-1} \in \mathbb{K}$, compute $P \in \mathbb{K}[x]_{<n}$ such that $P(a_0) = v_0, \ldots, P(a_{n-1}) = v_{n-1}$

Naive algorithm: Linear algebra, Vandermonde system $O(\mathsf{MM}(n))$

Lagrange's algorithm: Use $P(x) = \sum_{i=0}^{n-1} v_i \frac{\prod_{j \neq i}(x - a_j)}{\prod_{j \neq i}(a_i - a_j)}$ $O(n^2)$

Fast algorithm: Modified Lagrange formula

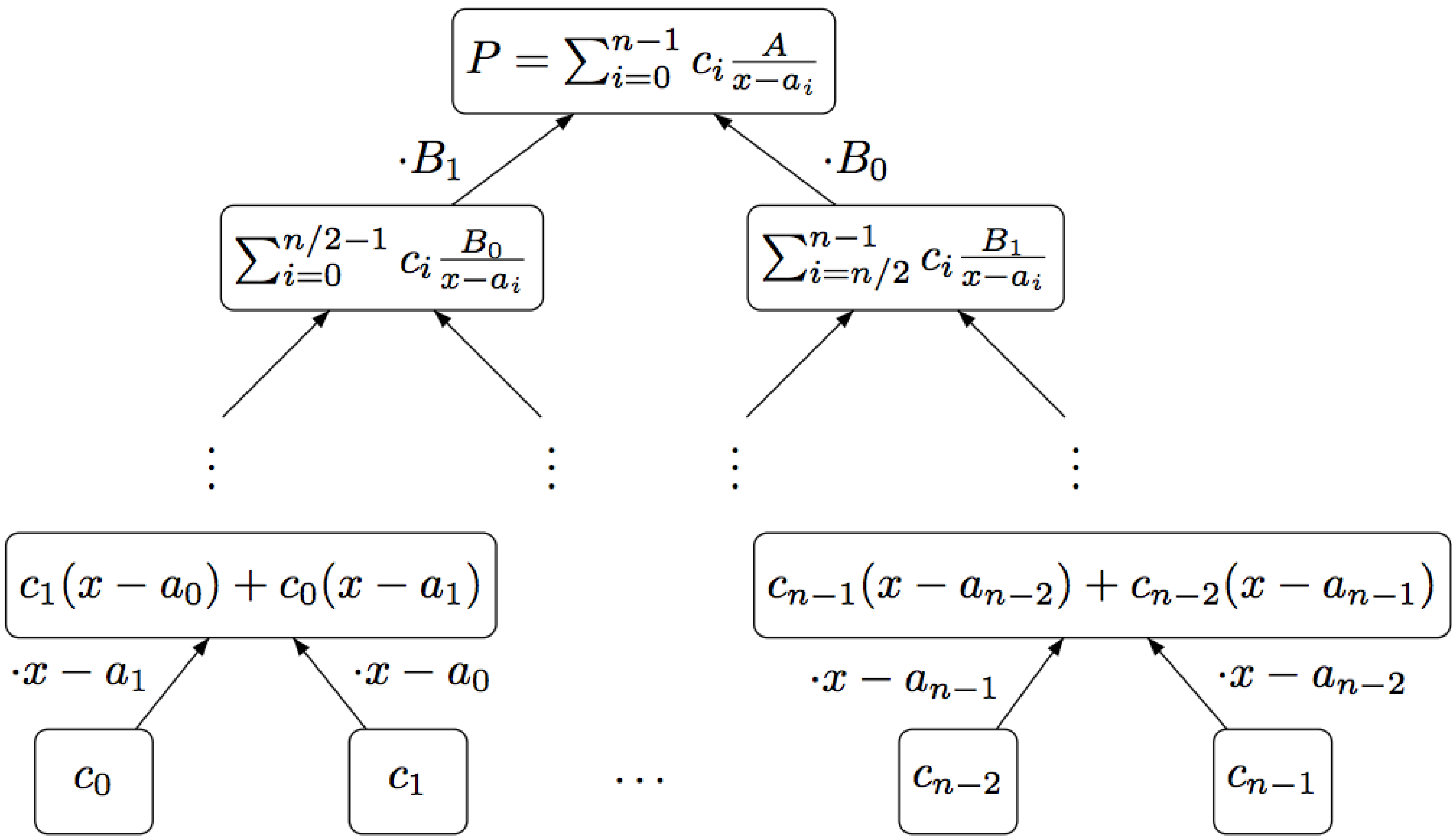
$$P = A(x) \cdot \sum_{i=0}^{n-1} \frac{v_i / A'(a_i)}{x - a_i}$$

- Compute $c_i = v_i / A'(a_i)$ by fast multipoint evaluation $O(\mathsf{M}(n) \log n)$
- Compute $\sum_{i=0}^{n-1} \frac{c_i}{x - a_i}$ by divide and conquer $O(\mathsf{M}(n) \log n)$

Fast interpolation

[Borodin-Moenck, 1974]

Problem: Given $a_0, \ldots, a_{n-1} \in \mathbb{K}$ mutually distinct, and $v_0, \ldots, v_{n-1} \in \mathbb{K}$, compute $P \in \mathbb{K}[x]_{<n}$ such that $P(a_0) = v_0, \ldots, P(a_{n-1}) = v_{n-1}$



Master Theorem: $\mathsf{C}(n) = 2 \cdot \mathsf{C}(n/2) + O(\mathsf{M}(n)) \implies \mathsf{C}(n) = O(\mathsf{M}(n) \log n)$

Decrease and conquer I

Evaluation-interpolation, geometric case

Subproduct tree, geometric case

[B-Schost, 2005]

Problem: Given $q \in \mathbb{K}$, compute $A = \prod_{i=0}^{n-1}(x - q^i)$

Idea: Compute $B_1 = \displaystyle\prod_{i=n/2}^{n-1}(x - q^i)$ from $B_0 = \displaystyle\prod_{i=0}^{n/2-1}(x - q^i)$, by a homothety

$$B_1(x) = B_0\left(\frac{x}{q^{n/2}}\right) \cdot q^{(n/2)^2}$$

Decrease and conquer:

- Compute $B_0(x)$ by a recursive call
- Deduce $B_1(x)$ from $B_0(x)$ $\qquad O(n)$
- Return $A(x) = B_0(x)B_1(x)$ $\qquad \mathsf{M}(n/2)$

Master Theorem: $\mathsf{C}(n) = \mathsf{C}(n/2) + O(\mathsf{M}(n)) \implies \mathsf{C}(n) = O(\mathsf{M}(n))$

Fast multipoint evaluation, geometric case

[Bluestein, 1970]

Problem: Given $q \in \mathbb{K}$ and $P \in \mathbb{K}[x]_{<n}$, compute $P(1), P(q), \ldots, P(q^{n-1})$

The needed values are: $$P(q^i) = \sum_{j=0}^{n-1} c_j q^{ij}, \qquad 0 \le i < n$$

Bluestein's trick: $$ij = \frac{(i+j)^2 - i^2 - j^2}{2} \implies q^{ij} = q^{(i+j)^2/2} \cdot q^{-i^2/2} \cdot q^{-j^2/2}$$

$$\implies \qquad P(q^i) = q^{-i^2/2} \cdot \underbrace{\sum_{j=0}^{n-1} c_j q^{-j^2/2} \cdot q^{(i+j)^2/2}}_{\text{convolution:}}$$

$$\left[x^{n-1+i}\right] \left(\sum_{k=0}^{n-1} c_k q^{-k^2/2} x^{n-k-1} \right) \left(\sum_{\ell=0}^{2n-2} q^{\ell^2/2} x^{\ell} \right)$$

Conclusion: Fast evaluation on a geometric sequence in $O(\mathsf{M}(n))$

Fast interpolation, geometric case

[B-Schost, 2005]

Problem: Given $q \in \mathbb{K}$, and $v_0, \ldots, v_{n-1} \in \mathbb{K}$, compute $P \in \mathbb{K}[x]_{<n}$ such that $P(1) = v_0, \ldots, P(q^{n-1}) = v_{n-1}$

Fast algorithm: Modified Lagrange formula

$$P = A(x) \cdot \sum_{i=0}^{n-1} \frac{v_i / A'(q^i)}{x - q^i}, \qquad A = \prod_i (x - q^i)$$

- Compute $\prod_{i=0}^{n-1} (x - q^i)$ by decrease and conquer $O(\mathsf{M}(n))$
- Compute $c_i = v_i / A'(q^i)$ by Bluestein's algorithm $O(\mathsf{M}(n))$
- Compute $\sum_{i=0}^{n-1} \frac{c_i}{x - q^i}$ by decrease and conquer $O(\mathsf{M}(n))$

Fast interpolation, geometric case

[B-Schost, 2005]

Problem: Given $q \in \mathbb{K}$, and $v_0, \ldots, v_{n-1} \in \mathbb{K}$, compute $P \in \mathbb{K}[x]_{<n}$ such that $P(1) = v_0, \ldots, P(q^{n-1}) = v_{n-1}$

Subproblem: Given $c_0, \ldots, c_{n-1} \in \mathbb{K}$, compute $\quad R(x) = \sum_{i=0}^{n-1} \frac{c_i}{x - q^i}$

Idea: change of representation – enough to compute $R \mod x^n$

Second idea: $R \bmod x^n =$ multipoint evaluation at $\{1, q^{-1}, \ldots, q^{-(n-1)}\}$:

$$\sum_{i=0}^{n-1} \frac{c_i}{x - q^i} \mod x^n = -\sum_{i=0}^{n-1} \left(\sum_{j=0}^{n-1} c_i q^{-i(j+1)} x^j \right) = -\sum_{j=0}^{n-1} C(q^{-j-1}) x^j$$

Conclusion: Algorithm for interpolation at a geometric sequence in $O(\mathsf{M}(n))$ (generalization of the IDFT)

Product of polynomial matrices

[B-Schost, 2005]

Problem: Given $A, B \in \mathcal{M}_r(\mathbb{K}[x]_{<n})$, compute $C = AB$

Idea: change of representation – evaluation-interpolation at a geometric sequence $\mathcal{G} = \{1, q, q^2, \ldots, q^{2n-2}\}$

- Evaluate A and B at $\mathcal{G}$ — $O(r^2\,\mathsf{M}(n))$
- Multiply values $C(v) = A(v)B(v)$ for $v \in \mathcal{G}$ — $O(n\,\mathsf{MM}(r))$
- Interpolate C from values — $O(r^2\,\mathsf{M}(n))$

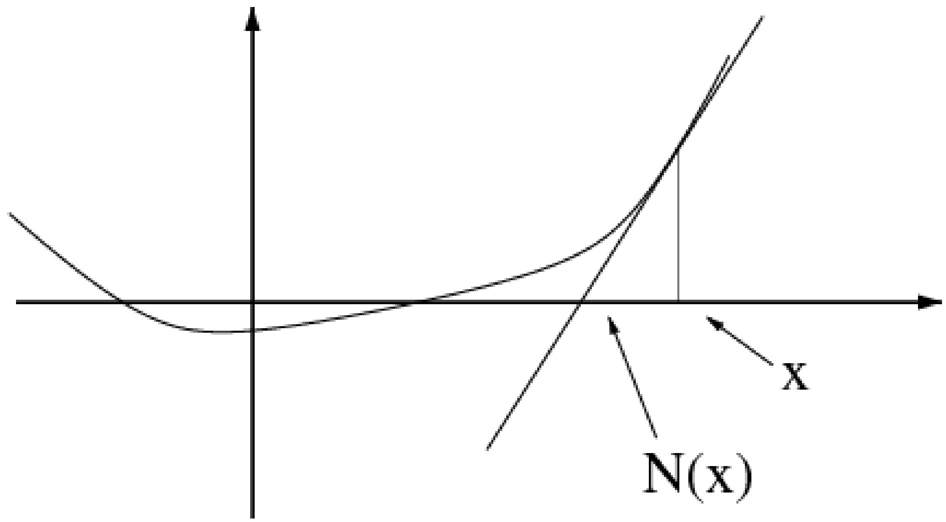
Total complexity — $O(r^2\,\mathsf{M}(n) + n\,\mathsf{MM}(r))$

Decrease and conquer II

Newton iteration

Newton's tangent method: real case

[Newton, 1671]



$$x_{\kappa+1} = \mathcal{N}(x_\kappa) = x_\kappa - (x_\kappa^2 - 2)/(2x_\kappa), \quad x_0 = 1$$

$$x_1 = 1.50000000000000000000000000000000$$

$$x_2 = 1.41666666666666666666666666666667$$

$$x_3 = 1.41421568627450980392156862745098$$

$$x_4 = 1.41421356237468991062629557889$01$$

$$x_5 = 1.41421356237309504880168896235025$$

Newton's tangent method: power series case

$$x_{\kappa+1} = \mathcal{N}(x_\kappa) = x_\kappa - (x_\kappa^2 - (1-t))/(2x_\kappa), \quad x_0 = 1$$

$$x_1 = 1 - \frac{1}{2}t$$

$$x_2 = 1 - \frac{1}{2}t - \frac{1}{8}t^2 - \frac{1}{16}t^3 - \frac{1}{32}t^4 - \frac{1}{64}t^5 - \frac{1}{128}t^6 - \frac{1}{256}t^7 - \frac{1}{512}t^8 - \frac{1}{1024}t^9 + \cdots$$

$$x_3 = 1 - \frac{1}{2}t - \frac{1}{8}t^2 - \frac{1}{16}t^3 - \frac{5}{128}t^4 - \frac{7}{256}t^5 - \frac{21}{1024}t^6 - \frac{33}{2048}t^7 - \frac{107}{8192}t^8 - \frac{177}{16384}t$$

Newton's tangent method: power series case

In order to solve $\varphi(x, g) = 0$ in $\mathbb{K}[[x]]$ (where $\varphi \in \mathbb{K}[[x, y]]$, $\varphi(0, 0) = 0$ and $\varphi_y(0, 0) \neq 0$), iterate

$$g_{\kappa+1} = g_\kappa - \frac{\varphi(g_\kappa)}{\varphi_y(g_\kappa)} \mod x^{2^{\kappa+1}}$$

$$g - g_{\kappa+1} = g - g_\kappa + \frac{\varphi(g) + (g_\kappa - g)\varphi_y(g) + O((g - g_\kappa)^2)}{\varphi_y(g) + O(g - g_\kappa)} = O((g - g_\kappa)^2).$$

- The number of correct coefficients doubles after each iteration
- Total cost $= \mathbf{2} \times \Big($ the cost of the last iteration $\Big)$

Theorem [Cook 1966, Sieveking 1972 & Kung 1974, Brent 1975]
Division, logarithm and exponential of power series in $\mathbb{K}[[x]]$ can be computed at precision N using $O(\mathsf{M}(N))$ operations in $\mathbb{K}$

Division, logarithm and exponential of power series

[Sieveking1972, Kung1974, Brent1975]

To compute the reciprocal of $f \in \mathbb{K}[[x]]$ with $f(0) \neq 0$, choose $\varphi(g) = 1/g - f$:

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

Complexity: $\mathsf{C}(N) = \mathsf{C}(N/2) + O(\mathsf{M}(N)) \quad \Longrightarrow \quad \mathsf{C}(N) = O(\mathsf{M}(N))$

Corollary: division of power series at precision N in $O(\mathsf{M}(N))$

Division, logarithm and exponential of power series

[Sieveking1972, Kung1974, Brent1975]

To compute the reciprocal of $f \in \mathbb{K}[[x]]$, choose $\varphi(g) = 1/g - f$:

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

Complexity: $\mathsf{C}(N) = \mathsf{C}(N/2) + O(\mathsf{M}(N)) \quad \Longrightarrow \quad \mathsf{C}(N) = O(\mathsf{M}(N))$

Corollary: division of power series at precision N in $O(\mathsf{M}(N))$

Corollary: Logarithm $\log(f) = -\sum_{i \geq 1} \frac{(1-f)^i}{i}$ of $f \in 1 + x\mathbb{K}[[x]]$ in $O(\mathsf{M}(N))$:

- compute the Taylor expansion of $h = f'/f$ modulo x^{N-1} $\quad O(\mathsf{M}(N))$
- take the antiderivative of h $\quad O(N)$

Division, logarithm and exponential of power series

[Sieveking1972, Kung1974, Brent1975]

To compute the reciprocal of $f \in \mathbb{K}[[x]]$, choose $\varphi(g) = 1/g - f$:

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

Complexity: $\mathsf{C}(N) = \mathsf{C}(N/2) + O(\mathsf{M}(N)) \quad \Longrightarrow \quad \mathsf{C}(N) = O(\mathsf{M}(N))$

Corollary: division of power series at precision N in $O(\mathsf{M}(N))$

Corollary: Logarithm $\log(f) = -\sum_{i \geq 1} \frac{(1-f)^i}{i}$ of $f \in 1 + x\mathbb{K}[[x]]$ in $O(\mathsf{M}(N))$:

- compute the Taylor expansion of $h = f'/f$ modulo x^{N-1} $\qquad O(\mathsf{M}(N))$
- take the antiderivative of h $\qquad O(N)$

Corollary: Exponential $\exp(f) = \sum_{i \geq 0} \frac{f^i}{i!}$ of $f \in x\mathbb{K}[[x]]$. Use $\phi(g) = \log(g) - f$:

$$g_0 = 1 \quad \text{and} \quad g_{\kappa+1} = g_\kappa - g_\kappa\left(\log(g_\kappa) - f\right) \quad \bmod x^{2^{\kappa+1}} \qquad \text{for } \kappa \geq 0.$$

Complexity: $\mathsf{C}(N) = \mathsf{C}(N/2) + O(\mathsf{M}(N)) \quad \Longrightarrow \quad \mathsf{C}(N) = O(\mathsf{M}(N))$

Application: Euclidean division for polynomials

[Strassen, 1973]

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

Naive algorithm: $O(N^2)$

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

Let $N = \deg(F)$ and $n = \deg(G)$. Then $\deg(Q) = N - n$, $\deg(R) < n$ and

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

Algorithm:

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

Application: extension of recurrences

[Shoup, 1991]

Problem: Given $N \in \mathbb{N}$ and the first n terms $u_0, \ldots, u_{n-1}$ of a recurrent sequence with constant coefficients of order n, compute $u_n, \ldots, u_N$

Naive algorithm: unroll the recurrence $O(N^2)$

Idea: $\sum_{i \geq 0} u_i x^i$ is rational $A(x)/B(x)$, with B given by the input recurrence, and $\deg(A) < \deg(B)$

Example (Fibonacci): $F_{i+2} = F_{i+1} + F_i \quad \Longleftrightarrow \quad \sum_i F_i x^i = \dfrac{F_0 + (F_1 - F_0)x}{1 - x - x^2}$

Algorithm:

- Compute A from B and $u_0, \ldots, u_{n-1}$ $O(\mathsf{M}(n))$
- Expand A/B modulo x^{N+1} $O(\mathsf{M}(N))$

Application: conversion coefficients $\leftrightarrow$ power sums

[Schönhage, 1982]

Any polynomial $F = x^n + a_1 x^{n-1} + \cdots + a_n$ in $\mathbb{K}[x]$ can be represented by its first n power sums $S_i = \sum_{F(\alpha)=0} \alpha^i$

Conversions coefficients $\leftrightarrow$ power sums can be performed

- either in $O(n^2)$ using Newton identities (naive way):

$$i a_i + S_1 a_{i-1} + \cdots + S_i = 0, \quad 1 \le i \le n$$

- or in $O(\mathsf{M}(n))$ using generating series

$$\frac{\mathsf{rev}(F)'}{\mathsf{rev}(F)} = -\sum_{i \ge 0} S_{i+1} x^i \quad \Longleftrightarrow \quad \mathsf{rev}(F) = \exp\left(-\sum_{i \ge 1} \frac{S_i}{i} x^i\right)$$

Application: special bivariate resultants

[B-Flajolet-Salvy-Schost, 2006]

Composed products and sums: manipulation of algebraic numbers

$$F \otimes G = \prod_{F(\alpha)=0, G(\beta)=0} (x - \alpha\beta), \quad F \oplus G = \prod_{F(\alpha)=0, G(\beta)=0} (x - (\alpha + \beta))$$

Output size: $N = \deg(F)\deg(G)$

Linear algebra: χ_{xy}, χ_{x+y} in $\mathbb{K}[x, y]/(F(x), G(y))$ $O(\mathsf{MM}(N))$

Resultants: $\mathrm{Res}_y\left(F(y), y^{\deg(G)} G(x/y)\right), \mathrm{Res}_y\left(F(y), G(x - y)\right)$ $O(N^{1.5})$

Better: $\otimes$ and $\oplus$ are easy in Newton representation $O(\mathsf{M}(N))$

$$\sum \alpha^s \sum \beta^s = \sum (\alpha\beta)^s \qquad \text{and}$$

$$\sum \frac{\sum(\alpha + \beta)^s}{s!} x^s = \left(\sum \frac{\sum \alpha^s}{s!} x^s\right)\left(\sum \frac{\sum \beta^s}{s!} x^s\right)$$

Corollary: Fast polynomial shift $P(x + a) = P(x) \oplus (x + a)$ $O(\mathsf{M}(\deg(P)))$

Newton iteration on power series: operators and systems

In order to solve an equation $\phi(Y) = 0$, with $\phi : (\mathbb{K}[[x]])^r \to (\mathbb{K}[[x]])^r$,

1. Linearize: $\phi(Y_\kappa - U) = \phi(Y_\kappa) - D\phi|_{Y_\kappa} \cdot U + O(U^2)$,
   where $D\phi|_Y$ is the differential of ϕ at Y.
2. Iterate: $Y_{\kappa+1} = Y_\kappa - U_{\kappa+1}$, where $U_{\kappa+1}$ is found by
3. Solve linear equation: $D\phi|_{Y_k} \cdot U = \phi(Y_\kappa)$ with $\operatorname{val} U > 0$.

Then, the sequence Y_κ converges quadratically to the solution Y.

Application: inversion of power series matrices

[Schulz, 1933]

To compute the inverse Z of a matrix of power series $Y \in \mathcal{M}_r(\mathbb{K}[[x]])$:

- Choose the map $\phi : Z \mapsto I - YZ$ with differential $D\phi|_Y : U \mapsto -YU$
- Equation for U: $-YU = I - YZ_\kappa \mod x^{2^{\kappa+1}}$
- Solution: $U = -Y^{-1}(I - YZ_\kappa) = -Z_\kappa(I - YZ_\kappa) \mod x^{2^{\kappa+1}}$

This yields the following Newton-type iteration for Y^{-1}

$$Z_{\kappa+1} = Z_\kappa + Z_\kappa(I_r - YZ_\kappa) \mod x^{2^{\kappa+1}}$$

Complexity:

$\mathsf{C}_{\text{inv}}(N) = \mathsf{C}_{\text{inv}}(N/2) + O(\mathsf{MM}(r, N)) \implies \mathsf{C}_{\text{inv}}(N) = O(\mathsf{MM}(r, N))$

Application: non-linear systems

In order to solve a system $Y = H(Y) = \phi(Y) + Y$, with $H : (\mathbb{K}[[x]])^r \to (\mathbb{K}[[x]])^r$, such that $I_r - \partial H/\partial Y$ is invertible at 0.

1. Linearize: $\phi(Y_\kappa - U) - \phi(Y_\kappa) = U - \partial H/\partial Y(Y_\kappa) \cdot U + O(U^2)$.
2. Iterate $Y_{\kappa+1} = Y_\kappa - U_{\kappa+1}$, where $U_{\kappa+1}$ is found by
3. Solve linear equation: $(I_r - \partial H/\partial Y(Y_\kappa)) \cdot U = H(Y_\kappa) - Y_\kappa$ with $\operatorname{val} U > 0$.

This yields the following Newton-type iteration:

$$\begin{cases} Z_{\kappa+1} & = Z_\kappa + Z_\kappa(I_r - (I_r - \partial H/\partial Y(Y_\kappa))Z_\kappa) \mod x^{2^{\kappa+1}} \\ Y_{\kappa+1} & = Y_\kappa - Z_{\kappa+1}(H(Y_\kappa) - Y_\kappa) \mod x^{2^{\kappa+1}} \end{cases}$$

computing simultaneously a matrix and a vector.

Application: quasi-exponential of power series matrices

[B-Chyzak-Ollivier-Salvy-Schost-Sedoglavic 2007]

To compute the solution $Y \in \mathcal{M}_r(\mathbb{K}[[x]])$ of the system $Y' = AY$

- choose the map $\phi : Y \mapsto Y' - AY$, with differential ϕ.
- the equation for U is $\quad U' - AU = Y_\kappa' - AY_\kappa \mod x^{2^{\kappa+1}}$
- the method of variation of constants yields the solution
  $U = Y_\kappa V_\kappa \bmod x^{2^{\kappa+1}}$, $\;Y_\kappa' - AY_\kappa = Y_\kappa V_\kappa' \bmod x^{2^{\kappa+1}}$

This yields the following Newton-type iteration for Y:

$$Y_{\kappa+1} = Y_\kappa - Y_\kappa \int Y_\kappa^{-1} \left(Y_\kappa' - AY_\kappa\right) \mod x^{2^{\kappa+1}}$$

Complexity:

$\mathsf{C}_{\text{solve}}(N) = \mathsf{C}_{\text{solve}}(N/2) + O(\mathsf{MM}(r, N)) \qquad \Longrightarrow \qquad \mathsf{C}_{\text{solve}}(N) = O(\mathsf{MM}(r, N))$