Fast Evaluation and Interpolation
&
Gcd and Extended Gcd

Alin Bostan

MPRI C-2-22
October 4, 2022

M2 Internship Projects

- *Algorithms for the parametrization of plane curves* (Bostan)
- *Algorithms for solving q-difference equations* (Bostan)
- *Multipoint power series expansions* (Lairez)
- *Univariate matrices for faster polynomial system solving* (Neiger)
- *Multi-level algebraic structures and faster guessing* (Neiger)

$\leadsto$ detailed descriptions available on request: contact us asap if interested!

The exercises from last week

(1) Let $\mathsf{T}(n)$ be the complexity of multiplication of $n \times n$ lower triangular matrices. Show that one can multiply any two $n \times n$ matrices in $O(\mathsf{T}(n))$ ops.

(2) Let $\mathbb{K}$ be a field, let $P \in \mathbb{K}[x]$ be of degree less than n and θ be a feasible exponent for matrix multiplication in $\mathcal{M}_n(\mathbb{K})$.

(a) Find an algorithm for the simultaneous evaluation of P at $\lceil \sqrt{n} \rceil$ elements of $\mathbb{K}$ using $O(n^{\theta/2})$ operations in $\mathbb{K}$.

(b) If Q is another polynomial in $\mathbb{K}[X]$ of degree less than n, show how to compute the first n coefficients of $P \circ Q := P(Q(x))$ in $O(n^{\frac{\theta+1}{2}})$ ops. in $\mathbb{K}$.

▷ Hint: Write $P(x)$ as $\sum_i P_i(x)(x^d)^i$, where d is well-chosen and the P_i's have degrees less than d.

Ex. 1

Let $\mathsf{T}(n)$ be the complexity of multiplication of $n \times n$ lower triangular matrices. Show that one can multiply any two $n \times n$ matrices in $O(\mathsf{T}(n))$ ops.

Solution:

▷ For any $n \times n$ matrices A and B,

$$\begin{bmatrix} 0 & 0 & 0 \\ B & 0 & 0 \\ 0 & A & 0 \end{bmatrix}^2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ AB & 0 & 0 \end{bmatrix}.$$

▷ Let α be a feasible exponent for multiplication of lower triangular matrices. Then, $n^\theta \leq \mathsf{T}(3n) = O(n^\alpha)$ and thus $\theta \leq \alpha$.

Ex. 2

Let $\mathbb{K}$ be a field, let $P \in \mathbb{K}[x]$ be of degree less than n and θ be a feasible exponent for matrix multiplication in $\mathcal{M}_n(\mathbb{K})$.

(a) Find an algorithm for the simultaneous evaluation of P at $\lceil \sqrt{n}\, \rceil$ elements of $\mathbb{K}$ using $O(n^{\theta/2})$ operations in $\mathbb{K}$.

(b) If Q is another polynomial in $\mathbb{K}[X]$ of degree less than n, show how to compute the first n coefficients of $P \circ Q := P(Q(x))$ in $O(n^{\frac{\theta+1}{2}})$ ops. in $\mathbb{K}$.

Solution 2(a):

▷ Write $P(x)$ as $\sum_i P_i(x)(x^d)^i$, where $d = \lceil \sqrt{n} \rceil$ and the P_i's have degrees $< d$

▷ Evaluations of the P_i's at the points $x_1, \ldots, x_d$ read off the matrix product

$$\begin{bmatrix} P_0(x_1) & \ldots & P_0(x_d) \\ \vdots & & \vdots \\ P_{d-1}(x_1) & \ldots & P_{d-1}(x_d) \end{bmatrix} = \begin{bmatrix} p_{0,0} & \ldots & p_{0,d-1} \\ \vdots & & \vdots \\ p_{d-1,0} & \ldots & p_{d-1,d-1} \end{bmatrix} \times \begin{bmatrix} 1 & \ldots & 1 \\ \vdots & & \vdots \\ x_1^{d-1} & \ldots & x_d^{d-1} \end{bmatrix}$$

Ex. 2

Solution 2(b): Baby step / giant step strategy

▷ Write $P(x)$ as $\sum_i P_i(x)(x^d)^i$, where $d = \lceil \sqrt{n} \rceil$ and the P_i's have degrees $< d$

▷ Compute $Q^2, \ldots, Q^d =: R$ and $R^2, \ldots, R^{d-1} \bmod x^n$ $\quad O(d\,\mathsf{M}(n)) = O(n^{\frac{\theta+1}{2}})$

For $p_{i,j} := [x^j]P_i$ and $q_{i,j} := [x^j]Q^i$ $(j < n, i < d)$, compute $P_i(Q) \bmod x^n$ using the $(d \times d) \times (d \times n)$ matrix product $\quad [x^j]P_i(Q) = \sum_k p_{i,k} q_{k,j}$

$$\begin{bmatrix} p_{0,0} & \cdots & p_{0,d-1} \\ \vdots & & \vdots \\ p_{d-1,0} & \cdots & p_{d-1,d-1} \end{bmatrix} \times \begin{bmatrix} q_{0,0} & \cdots & q_{0,n-1} \\ \vdots & & \vdots \\ q_{d-1,0} & \cdots & q_{d-1,n-1} \end{bmatrix},$$

▷ Can be done using $\lceil n/d \rceil = O(d)$ products of $d \times d$ matrices $\quad O(d^{\theta+1})$

▷ Final recombination $P(Q) \bmod x^n = \sum_{i=0}^{d-1} P_i(Q) R^i \bmod x^n$ $\quad O(d\,\mathsf{M}(n))$

Context

▷ Main concepts: Evaluation-interpolation paradigm and Modular algorithms

▷ Alternative representations of algebraic objects: e.g., polynomials given

- by list of coefficients: useful for fast division
- by list of values taken on given points: useful for fast multiplication (FFT)

▷ Modular algorithms based on fast conversions between representations, e.g. evaluation-interpolation, Chinese Remaindering

▷ Avoid intermediate expression swell, e.g. det of polynomial matrices

▷ Important issue: choice of the moduli (evaluation points), e.g. fast factorial

Main problems and results

Multipoint evaluation Given P in $\mathbb{A}[X]$, of degree $< n$, compute the values

$$P(a_0), \ldots, P(a_{n-1}).$$

Interpolation Given $v_0, \ldots, v_{n-1} \in \mathbb{A}$, with $a_i - a_j$ invertible in $\mathbb{A}$ if $i \neq j$, find the polynomial $P \in \mathbb{A}[X]$ of degree $< n$ such that

$$P(a_0) = v_0, \ldots, P(a_{n-1}) = v_{n-1}.$$

Theorem One can solve both problems in:

- $O(\mathsf{M}(n) \log n)$ ops. in $\mathbb{A}$
- $O(\mathsf{M}(n))$ ops. in $\mathbb{A}$ if the a_i's are in geometric progression

▷ Extension to fast polynomial/integer Chinese remaindering

Waring-Lagrange interpolation

THEOREM I.

Aſſume an equation $a+bx+cx^2+dx^3 \ldots\ldots x^{n-1}=y$, in which the co-efficients a, b, c, d, e, &c. are invariable; let α, β, γ, δ, ε, &c. denote n values of the unknown quantity x, whoſe correſpondent values of y let be repreſented by s^α, s^β, s^γ, s^δ, s^ε, &c. Then will the equation $a+bx+cx^2+dx^3+ex^4 \ldots x^{n-1}=y=$

$$\frac{\overline{x-\beta}\times\overline{x-\gamma}\times\overline{x-\delta}\times\overline{x-\varepsilon}\times \&c.}{\overline{\alpha-\beta}\times\overline{\alpha-\gamma}\times\overline{\alpha-\delta}\times\overline{\alpha-\varepsilon}\times \&c.}\times s^\alpha+\frac{\overline{x-\alpha}\times\overline{x-\gamma}\times\overline{x-\delta}\times\overline{x-\varepsilon}\times \&c.}{\overline{\beta-\alpha}\times\overline{\beta-\gamma}\times\overline{\beta-\delta}\times\overline{\beta-\varepsilon}\times \&c.}\times s^\beta$$

$$+\frac{\overline{x-\alpha}\times\overline{x-\beta}\times\overline{x-\delta}\times\overline{x-\varepsilon}\times \&c.}{\overline{\gamma-\alpha}\times\overline{\gamma-\beta}\times\overline{\gamma-\delta}\times\overline{\gamma-\varepsilon}\times \&c.}\times s^\gamma+\frac{\overline{x-\alpha}\times\overline{x-\beta}\times\overline{x-\gamma}\times\overline{x-\varepsilon}\times \&c.}{\overline{\delta-\alpha}\times\overline{\delta-\beta}\times\overline{\delta-\gamma}\times\overline{\delta-\varepsilon}\times \&c.}\times s^\delta$$

$$+\frac{\overline{x-\alpha}\times\overline{x-\beta}\times\overline{x-\gamma}\times\overline{x-\delta}\times \&c.}{\overline{\varepsilon-\alpha}\times\overline{\varepsilon-\beta}\times\overline{\varepsilon-\gamma}\times\overline{\varepsilon-\delta}\times \&c.}\times s^\varepsilon+\&c.$$

[Waring, 1779 – “Problems concerning Interpolations”] qu'en faisant $x = p$ on ait

$$A = 1, \quad B = 0, \quad C = 0, \quad \ldots;$$

que de même, en faisant $x = q$, on ait

$$A = 0, \quad B = 1, \quad C = 0, \quad D = 0, \quad \ldots;$$

qu'en faisant $x = r$, on ait pareillement

$$A = 0, \quad B = 0, \quad C = 1, \quad D = 0, \quad \ldots, \quad \text{etc.};$$

d'où il est facile de conclure que les valeurs de A, B, C, ... doivent être de cette forme

$$A = \frac{(x-q)(x-r)(x-s)\ldots}{(p-q)(p-r)(p-s)\ldots},$$

$$B = \frac{(x-p)(x-r)(x-s)\ldots}{(q-p)(q-r)(q-s)\ldots},$$

$$C = \frac{(x-p)(x-q)(x-s)\ldots}{(r-p)(r-q)(r-s)\ldots},$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots,$$

en prenant autant de facteurs, dans les numérateurs et dans les dénominateurs, qu'il y aura de points donnés de la courbe, moins un.

Cette dernière expression de y, quoique sous une forme différente, revient cependant au même, comme on peut s'en assurer par le calcul, en développant les valeurs des quantités Q_1, R_2, S_3, ..., et ordonnant les termes suivant les quantités P, Q, R, ...; mais elle est préférable par la simplicité de l'Analyse sur laquelle elle est fondée, et par sa forme même, qui est beaucoup plus commode pour le calcul.

[Lagrange, 1795 – "Sur l'usage des courbes dans la solution des problèmes"]

Fast polynomial division

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:

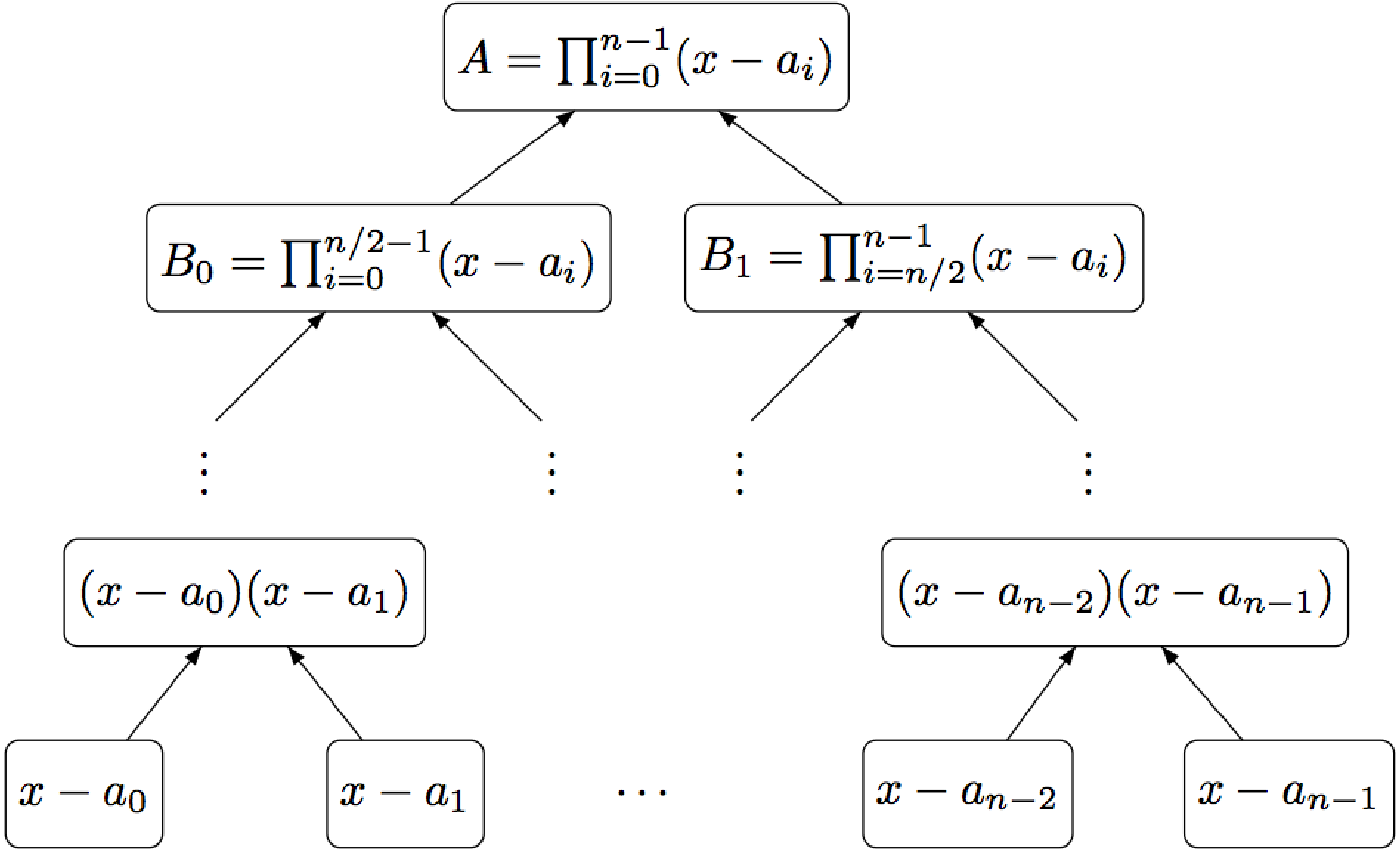
- Compute $\mathrm{rev}(Q) = \mathrm{rev}(F)/\mathrm{rev}(G) \mod x^{N-n+1}$ $O(\mathsf{M}(N))$
- Recover Q $O(1)$
- Deduce $R = F - QG$ $O(\mathsf{M}(N))$

Evaluation-interpolation, general case

Subproduct tree

[Horowitz, 1972]

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



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

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 $P(a_0), \ldots, P(a_{n-1})$

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

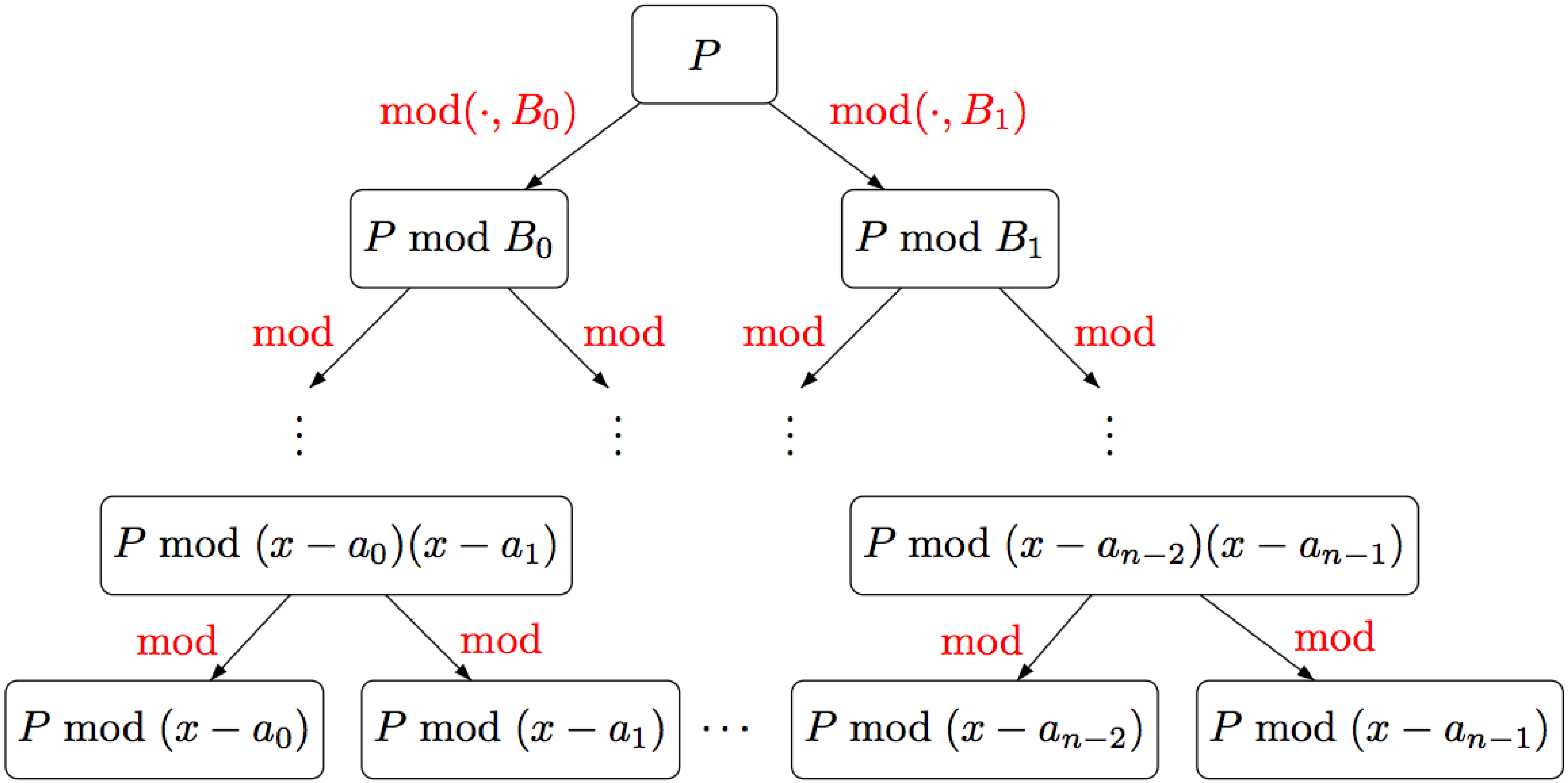
- $P_0 := P \mod \underbrace{(x-a_0)\cdots(x-a_{n/2-1})}_{B_0}$
- $P_1 := P \mod \underbrace{(x-a_{n/2})\cdots(x-a_{n-1})}_{B_1}$

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

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 $P(a_0), \ldots, P(a_{n-1})$



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

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: Based on the "modified Lagrange formula"

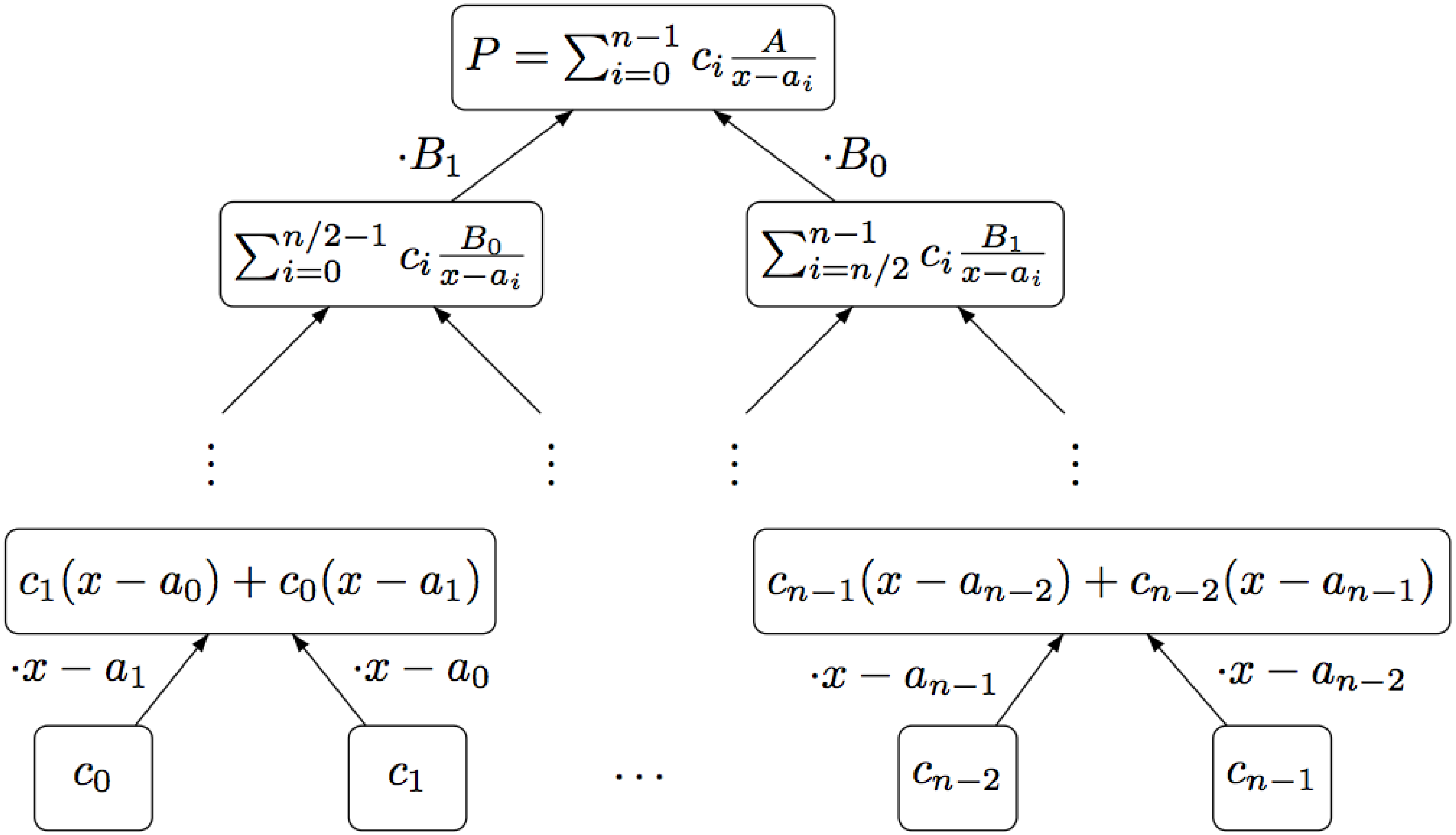
$$P(x) = A(x) \cdot \sum_{i=0}^{n-1} \frac{v_i / A'(a_i)}{x - a_i}$$

- 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 $P(a_0) = v_0, \ldots, P(a_{n-1}) = v_{n-1}$



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

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

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

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

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

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

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

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 FFT algorithm computing the IDFT)

Product of polynomial matrices

[B.-Schost, 2005]

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

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

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

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

An exercise for next Tuesday

Let f and g be two polynomials in $\mathbb{K}[x, y]$ of degrees at most d_x in x and at most d_y in y.

(a) Show that it is possible to compute the product $h = fg$ using

$$O(\mathsf{M}(d_x d_y))$$

arithmetic operations in $\mathbb{K}$.
Hint: Use the substitution $x \leftarrow y^{2d_y+1}$ to reduce the problem to the product of univariate polynomials.

(b) Improve this result by proposing an evaluation-interpolation scheme which allows the computation of h in

$$O(d_x\,\mathsf{M}(d_y) + d_y\,\mathsf{M}(d_x))$$

arithmetic operations in $\mathbb{K}$.

1. Space-saving versions

	Time	Space	Reference
multipoint evaluation size-n polynomial on n points	$3/2\mathsf{M}(n)\log(n)$	$n\log(n)$	[2]
	$7/2\mathsf{M}(n)\log(n)$	n	[7], Lemma 3.1
	$(4+2\lambda_s/\log(\frac{c_s+3}{c_s+2}))\mathsf{M}(n)\log(n)$	$O(1)$	Theorem 3.4
interpolation size-n polynomial on n points	$5/2\mathsf{M}(n)\log(n)$	$n\log(n)$	[2]
	$5\mathsf{M}(n)\log(n)$	$2n$	[6, 7], Lemma 3.3
	$\simeq 105\mathsf{M}(n)\log(n)$	$O(1)$	Theorem 3.6

[Giorgi, Grenet & Roche, ISSAC, 2020]

[2] A. Bostan, G. Lecerf, and É. Schost. 2003. Tellegen's Principle into Practice. In ISSAC'03, 37–44.

[6] J. von zur Gathen and V. Shoup. 1992. Computing Frobenius maps and factoring polynomials. Comput. Complex. 2, 3 (1992), 187–224.

[7] P. Giorgi, B. Grenet, and D. S. Roche. 2019. Generic reductions for in-place polynomial multiplication. In ISSAC'19, 187–194.

2. More general evaluation and interpolation

SIAM J. Comput.
Vol. 5, No. 4, December 1976

A GENERALIZED ASYMPTOTIC UPPER BOUND ON FAST POLYNOMIAL EVALUATION AND INTERPOLATION*

FRANCIS Y. CHIN†

Abstract. It is shown in this paper that the evaluation and interpolation problems corresponding to a set of points, $\{x_i\}_{i=0}^{n-1}$, with $(c_i - 1)$ higher derivatives at each x_i such that $\sum_{i=1}^{n-1} c_i = N$, can be solved in $O([N \log N][(\log n) + 1])$ steps.[1] This upper bound matches perfectly with the known upper bounds of the two extreme cases, which are $O(N \log^2 N)$ and $O(N \log N)$ steps when $n = N$ and $n = 1$, respectively.

Key words. polynomial evaluation, polynomial interpolation, asymptotic upper bounds

	Evaluation	Interpolation
(a) N points	$O(N \log^2 N)$ [6], [5]	$O(N \log^2 N)$ [6], [5]
(b) n points, $\{x_i\}_{i=0}^{n-1}$ and their corresponding $(c_i - 1)$ derivatives such that $\sum_{i=0}^{n-1} c_i = N$	$O([N \log N][(\log n) + 1])$ (Theorem 1)	$O([N \log N][(\log n) + 1])$ (Theorem 2)
(c) Single point and all its derivatives	$O(N \log N)$ [2], [9]	$O(N \log N)$ [2], [9]

[Chin, SIAM J. Comput., 1976]

3. Multivariate sparse interpolation

Q.-L. Huang, X.-S. Gao / Journal of Symbolic Computation 101 (2020) 367–386

Table 1
A "soft-Oh" comparison for SLP polynomials over an arbitrary ring $\mathcal{R}$.

Algorithms	Total Cost	Type
Dense	LD^n	Deterministic
Garg and Schost (2009)	$Ln^2T^4\log^2 D$	Deterministic
Randomized (Giesbrecht and Roche, 2011)	$Ln^2T^3\log^2 D$	Las Vegas
Arnold et al. (2013)	$Ln^3T\log^3 D$	Monte Carlo
This paper (Theorem 5.8)	$Ln^2T^2\log^2 D + LnT\log^3 D$	Deterministic
This paper (Theorem 6.8)	$LnT\log^3 D$	Monte Carlo

Table 2
A "soft-Oh" comparison for SLP polynomials over finite field $\mathbb{F}_q$.

Algorithms	Bit Complexity	Algorithm type
Garg and Schost (2009)	$Ln^2T^4\log^2 D\log q$	Deterministic
Randomized Garg-Schost (Giesbrecht and Roche, 2011)	$Ln^2T^3\log^2 D\log q$	Las Vegas
Giesbrecht and Roche (2011)	$Ln^2T^2\log^2 D(n\log D+\log q)$	Las Vegas
Arnold et al. (2013)	$Ln^3T\log^3 D\log q$	Monte Carlo
Arnold et al. (2014)	$LnT\log^2 D(\log D+\log q)+n^{\omega}T$	Monte Carlo
Arnold et al. (2016)	$Ln\log D(T\log D+n)(\log D+\log q)$ $+n^{\omega-1}T\log D+n^{\omega}\log D$	Monte Carlo
This paper (Theorem 5.8)	$Ln^2T^2\log^2 D\log q+LnT\log^3 D\log q$	Deterministic
This paper (Theorem 6.8)	$LnT\log^3 D\log q$	Monte Carlo
This paper (Theorem 6.11)	$LnT\log^2 D(\log q+\log D)$	Monte Carlo

[Huang & Gao, JSC, 2020]

GCD and Extended GCD

GCD

If $A, B \in \mathbb{K}[x]$, then $G \in \mathbb{K}[x]$ is a gcd of A and B if

- G divides both A and B,
- any common divisor of A and B divides G.

▷ It is a generator of the ideal of $\mathbb{K}[x]$ generated by A and B, i.e.,

$$\left\{U \cdot A + V \cdot B \,\middle|\, U, V \in \mathbb{K}[x]\right\} = \left\{W \cdot G \,\middle|\, W \in \mathbb{K}[x]\right\}$$

▷ In terms of roots: $Z(\mathrm{gcd}(A, B)) = Z(A) \cap Z(B)$

▷ It is unique up to a constant: the gcd, after normalization (G monic)

▷ It is useful for:

- normalization (simplification) of rational functions
- squarefree factorization of univariate polynomials

▷ Computation: Euclidean algorithm

Euclidean algorithm

Euclid(A, B)

Input A and B in $\mathbb{K}[x]$.

Output A gcd G of A and B.

1. $R_0 := A$; $R_1 := B$; $i := 1$.
2. While R_i is non-zero, do:

   $R_{i+1} := R_{i-1} \bmod R_i$

   $i := i + 1$.
3. Return R_{i-1}.

▷ Correctness: $\gcd(F, G) = \gcd(G, F \bmod G)$

▷ Termination: $\deg(B) > \deg(R_2) > \deg(R_1) > \cdots$

▷ Quadratic complexity: $O\big(\deg(A)\deg(B)\big)$ operations in $\mathbb{K}$

Extended GCD

If $A, B \in \mathbb{K}[x]$, then $G = \gcd(A, B)$ satisfies (Bézout relation)

$$G = U \cdot A + V \cdot B, \quad \text{with } U, V \in \mathbb{K}[x]$$

▷ The co-factors U and V are unique if one further asks

$$\deg(U) < \deg(B) - \deg(G) \quad \text{and} \quad \deg(V) < \deg(A) - \deg(G)$$

Then one calls (G, U, V) the extended gcd of A and B.

▷ Example: for $A = a + bx$ with $a \neq 0$ and $B = 1 + x^2$,

$$G = 1 \quad \text{and} \quad \frac{a - bx}{a^2 + b^2} \cdot A + \frac{b^2}{a^2 + b^2} \cdot B = 1$$

Extended GCD

Usefulness of Bézout coefficients:

- modular inversion and division in a quotient ring $Q = \mathbb{K}[x]/(B)$: A is invertible in Q if and only if $\gcd(A, B) = 1$. In this case: the inverse of A in Q is equal to U, where $U \cdot A + V \cdot B = 1$.
- Lecture 5 (18/10): proof of "Any algebraic function is D-finite"

▷ Example: For $A = a + bx, B = 1 + x^2$, the inverse of A mod B is

$$U = \frac{a - bx}{a^2 + b^2}.$$

▷ Computation: Extended Euclidean algorithm

Extended Euclidean algorithm

ExtendedEuclid(A, B)

Input A and B in $\mathbb{K}[x]$.

Output A gcd G of A and B, and cofactors U and V.

1. $R_0 := A$; $U_0 := 1$; $V_0 := 0$; $R_1 := B$; $U_1 := 0$; $V_1 := 1$; $i := 1$.
2. While R_i is non-zero, do:
   (a) $(Q_i, R_{i+1}) := \mathrm{QuotRem}(R_{i-1}, R_i)$ # $R_{i-1} = Q_i R_i + R_{i+1}$
   (b) $U_{i+1} := U_{i-1} - Q_i U_i$; $V_{i+1} := V_{i-1} - Q_i V_i$.
   (c) $i := i + 1$.
3. Return $\big(R_{i-1}, U_{i-1}, V_{i-1}\big)$.

▷ Correctness: $R_i = U_i A + V_i B$ (by induction):

$$R_{i+1} = R_{i-1} - Q_i R_i = U_{i-1} A + V_{i-1} B - Q_i (U_i A + V_i B) = U_{i+1} A + V_{i+1} B$$

▷ Quadratic complexity: $O\big(\deg(A)\deg(B)\big)$ operations in $\mathbb{K}$

LCM

If $A, B \in \mathbb{K}[x]$, then $L \in \mathbb{K}[x]$ is an lcm of A and B if

- both A and B divide L,
- any common multiple of A and B is divisible by L.

▷ It is a generator of the ideal $(A) \cap (B)$ of $\mathbb{K}[x]$, i.e.,

$$\left\{U \cdot A = V \cdot B \;\middle|\; U, V \in \mathbb{K}[x]\right\} = \left\{W \cdot L \;\middle|\; W \in \mathbb{K}[x]\right\}$$

▷ In terms of roots: $Z(\mathrm{lcm}(A, B)) = Z(A) \cup Z(B)$

▷ It is unique up to a constant: the lcm, after normalization (L monic)

▷ Computation: either using the formula $\mathrm{lcm}(A, B) = AB/\gcd(A, B)$, or by the half-extended Euclidean algorithm

Half-Extended Euclidean algorithm

HalfExtendedEuclid(A, B)

Input: A and B in $\mathbb{K}[x]$.

Output: A gcd G and an lcm L of A and B.

1. $R_0 := A$; $U_0 := 1$; $R_1 := B$; $U_1 := 0$; $i := 1$.
2. While R_i is non-zero, do:
   (a) $(Q_i, R_{i+1}) := \mathrm{QuotRem}(R_{i-1}, R_i)$ $\quad\quad \# R_{i-1} = Q_i R_i + R_{i+1}$
   (b) $U_{i+1} := U_{i-1} - Q_i U_i$.
   (c) $i := i + 1$.
3. Return $\big(R_{i-1}, U_i A\big)$.

▷ Quadratic complexity: $O\big(\deg(A)\deg(B)\big)$ operations in $\mathbb{K}$