1. Polynomials and Integers

CS-E4500 Advanced Course on Algorithms
Spring 2020

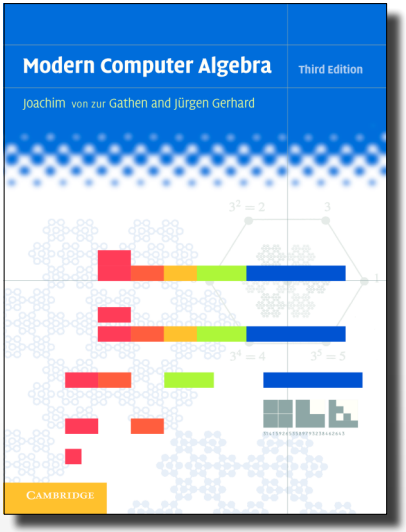
Petteri Kaski
Department of Computer Science
Aalto University

Key content for Lecture 1

- A boot camp of basic concepts and definitions in algebra
- Polynomials in one variable (univariate polynomials)
- Basic tasks and first algorithms for univariate polynomials
  - addition
  - multiplication
  - division (quotient and remainder)
  - evaluation
  - interpolation
  - greatest common divisor
- Evaluation–interpolation -duality of polynomials
- The (traditional) extended Euclidean algorithm and its analysis

A boot camp of basic concepts and definitions in algebra

(von zur Gathen and Gerhard [10],
Sections 2.2–3.2, 25.1–4)



Group

- A **group** is a nonempty set G with a binary operation $\cdot : G \times G \to G$ satisfying
  1. for all $a, b, c \in G$ we have $(a \cdot b) \cdot c = a \cdot (b \cdot c)$, (Associativity)
  2. there exists a $1 \in G$ such that $a \cdot 1 = 1 \cdot a = a$ for all $a \in G$, (Identity)
  3. for all $a \in G$ there exists an $a^{-1} \in G$ with $a \cdot a^{-1} = a^{-1} \cdot a = 1$ (Inverses)
- A group G is **commutative** if for all $a, b \in G$ we have $a \cdot b = b \cdot a$
- *Examples*:
  - $(\mathbb{Z}, +, 0)$ and $(\mathbb{Z}_m, +, 0)$ for $m \in \mathbb{Z}_{\geq 2}$ are commutative groups
  - $(\mathbb{Q} \setminus \{0\}, \cdot, 1)$ and $(\mathbb{Z}_m^\times, \cdot, 1)$ for $\mathbb{Z}_m^\times = \{1 \leq a < m : \gcd(a, m) = 1\}$ are commutative groups

(Commutative) ring

- A **ring** R is a set with two binary operations $+ : R \times R \to R$ and $\cdot : R \times R \to R$ satisfying
  1. R together with $+$ is a commutative group with identity 0,
  2. $\cdot$ is associative,
  3. R has an identity element 1 for $\cdot$,
  4. for all $a, b, c \in R$ we have $a(b + c) = (ab) + (ac)$ and $(b + c)a = (ba) + (ca)$
- A ring R is **commutative** if $\cdot$ is commutative
- A ring R is **nontrivial** if $0 \neq 1$
- Unless mentioned otherwise, in what follows we always assume that a ring R is both commutative and nontrivial
- *Examples*:
  $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$, $\mathbb{Z}_m$ for $m \in \mathbb{Z}_{\geq 2}$

Example: $\mathbb{Z}_5$ (the integers modulo 5)

- One way to represent a (finite) ring is to give the addition and multiplication tables for the operations operations $+$ and $\cdot$
- In the two tables below, the entries at row x column y are $x+y$ and $x \cdot y$, respectively

$+$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

$\cdot$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

(1)

Example: $\mathbb{Z}_6$ (the integers modulo 6)

- Below are the addition and multiplication tables for $\mathbb{Z}_6$

$+$	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

$\cdot$	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

(2)

- Compare the *multiplication* tables for $\mathbb{Z}_6$ (above) and $\mathbb{Z}_5$ (see (1))
  — what qualitative differences can you spot?

Example: $\mathbb{Z}_{10}$ (the integers modulo 10)

- Here is a yet further example, the integers modulo 10

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	0
2	2	3	4	5	6	7	8	9	0	1
3	3	4	5	6	7	8	9	0	1	2
4	4	5	6	7	8	9	0	1	2	3
5	5	6	7	8	9	0	1	2	3	4
6	6	7	8	9	0	1	2	3	4	5
7	7	8	9	0	1	2	3	4	5	6
8	8	9	0	1	2	3	4	5	6	7
9	9	0	1	2	3	4	5	6	7	8

·	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	0	2	4	6	8
3	0	3	6	9	2	5	8	1	4	7
4	0	4	8	2	6	0	4	8	2	6
5	0	5	0	5	0	5	0	5	0	5
6	0	6	2	8	4	0	6	2	8	4
7	0	7	4	1	8	5	2	9	6	3
8	0	8	6	4	2	0	8	6	4	2
9	0	9	8	7	6	5	4	3	2	1

(3)

- What patterns can you identify from the multiplication table?

Field, unit, associate

- A **unit** in a ring R is an element $u \in R$ for which there exists a multiplicative inverse $v \in R$ with $uv = 1$
- The set $R^\times$ of all units of R is a group under multiplication
- A ring R is a **field** if all nonzero elements of R are units
- *Examples:* (of fields)
  $\mathbb{Q}, \mathbb{R}, \mathbb{C}, \mathbb{Z}_p$ for p prime
- We say that $a \in R$ is an **associate** of $b \in R$ and write $a \sim b$ if there exists a unit $u \in R$ such that $a = ub$
- $\sim$ is an equivalence relation on R

Examples / work points

- Study the multiplication table for $\mathbb{Z}_5$ in (1)
  — how can you identify which elements are units?
- Based on the units that you identify, conclude that $\mathbb{Z}_5$ is a field
- By studying the multiplication table for $\mathbb{Z}_6$ in (2), conclude that $\mathbb{Z}_6$ is *not* a field by identifying a nonzero element in $\mathbb{Z}_6$ that does not have a multiplicative inverse
- Study (2) and (3). Which elements are units in $\mathbb{Z}_6$? How about in $\mathbb{Z}_{10}$?
- Determine the equivalence classes for the associate relation $\sim$ in $\mathbb{Z}_5$, $\mathbb{Z}_6$, and $\mathbb{Z}_{10}$

Polynomials over a ring (1/2)

- Let R be a ring and let x be a formal indeterminate
- A **polynomial** $a \in R[x]$ in x over R is a finite sequence $(\alpha_0, \alpha_1, \ldots, \alpha_n)$ of elements of R (the **coefficients** of a) which we write as

$$a = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \ldots + \alpha_{n-1} x^{n-1} + \alpha_n x^n = \sum_{i=0}^{n} \alpha_i x^i$$

- A polynomial a is **nonzero** if there exists a $j = 0, 1, \ldots, n$ with $\alpha_j \neq 0$
- For nonzero a, we assume that $\alpha_n \neq 0$ and say that $n = \deg a$ is the **degree** of a; the coefficient $\alpha_n = \mathrm{lc}(a)$ is the **leading coefficient** of a
- For zero a, it is convenient to assume that $a = (0)$ and set $\deg a = -\infty$
- A nonzero polynomial is **monic** if $\mathrm{lc}(a) = 1$

Polynomials over a ring (2/2)

- The set $R[x]$ equipped with the usual notions of addition and multiplication of polynomials (recalled in what follows) is a ring with additive identity (0) and multiplicative identity (1) for $0, 1 \in R$
- As a notational convention when working with polynomials, we use symbols x, y, z, w late in the Roman alphabet for formal indeterminates, and symbols $a, b, c, \ldots, s, t$ early in the Roman alphabet for polynomials
- We use symbols $\alpha, \beta, \gamma, \ldots, \omega$ in the Greek alphabet for elements in R

Complexity of an algorithm

- When studying algorithms that compute with given elements of $R[x]$, we adopt the convention of counting the number of **arithmetic operations** in R as a measure of the "running time" of an algorithm
- Arithmetic operations in R include addition, subtraction, multiplication and taking a multiplicative inverse (of a unit)
- We focus on **worst-case running time** (worst-case number of arithmetic operations in R) as a function of the degree(s) of the input polynomial(s) in $R[x]$
- We will work with asymptotic notation $O(\ )$ and $\tilde{O}(\ )$

Addition of polynomials

- Let $a = \sum_i \alpha_i x^i$, $b = \sum_i \beta_i x^i \in R[x]$ be given as input with $\deg a = n$ and $\deg b = m$
- The sum $c = a + b = \sum_i \gamma_i x^i \in R[x]$ is the polynomial with $\deg c \leq \max(n, m)$ defined for all $i = 0, 1, \ldots, \max(n, m)$ by

$$\gamma_i = \alpha_i + \beta_i \in R$$

- Given a, b as input, it is immediate that we can compute c in $O(\max(n, m))$ operations in R
- Subtraction and multiplication with a given element of R are defined analogously

Multiplication of polynomials

- Let $a = \sum_i \alpha_i x^i$, $b = \sum_i \beta_i x^i \in R[x]$ be given as input with $\deg a = n$ and $\deg b = m$
- The product $c = ab = \sum_i \gamma_i x^i \in R[x]$ is the polynomial with $\deg c \leq n + m$ defined for all $i = 0, 1, \ldots, n + m$ by

$$\gamma_i = \sum_{j=0}^{i} \alpha_j \beta_{i-j} \in R$$

- Given a, b as input, it is immediate that we can compute c in $O((n + m)^2)$ operations in R
- ... but could we do better? The output consists of only $O(n + m)$ elements of R ...

Polynomial division (quotient and remainder)

- Let $a = \sum_i \alpha_i x^i$, $b = \sum_i \beta_i x^i \in R[x]$ be given as input with $\deg a = n$, $\deg b = m$, $n \geq m \geq 0$, and suppose that $\beta_m \in R$ is a unit
- We want to compute $q, r \in R[x]$ with $a = qb + r$ and $\deg r < m$
- The classical division algorithm:
  1. $r \leftarrow a$, $\mu \leftarrow \beta_m^{-1}$
  2. **for** $i = n - m, n - m - 1, \ldots, 0$ **do**
  3. **if** $\deg r = m + i$ **then** $\eta_i \leftarrow \mathrm{lc}(r)\mu$, $r \leftarrow r - \eta_i x^i b$
     else $\eta_i \leftarrow 0$
  4. **return** $q = \sum_{i=0}^{n-m} \eta_i x^i$ and r
- We leave checking that $a = qb + r$ and $\deg r < m$ as an exercise; given a, b as input, it is immediate that we can compute q, r in $O((n + m)^2)$ operations in R
- ... but could we do better? The output consists of only $O(n + m)$ elements of R ...

Example (quotient and remainder)

- $a = x^4 + x^3 + x^2 + 1 \in \mathbb{Z}_2[x]$, $b = x^2 + 1 \in \mathbb{Z}_2[x]$
- $n = 4$, $m = 2$
- $\mu = \beta_m^{-1} = 1^{-1} = 1 \in \mathbb{Z}_2$
- Tracing the **for**-loop for $i = n - m, n - m - 1, \ldots, 0$, we have

i	η_i	r
		$x^4 + x^3 + x^2 + 1$
2	1	$x^3 + 1$
1	1	$x + 1$
0	0	$x + 1$

- $q = \eta_2 x^2 + \eta_1 x + \eta_0 = x^2 + x$, $r = x + 1$

Evaluation (at a single point)

- Let $a = \sum_i \alpha_i x^i \in R[x]$ and $\xi \in R$ be given as input with $\deg a = n$
- We want to compute $a(\xi) = \sum_{i=0}^{n} \alpha_i \xi^i \in R$
- **Horner's rule**:

$$a(\xi) = (\cdots(((\alpha_n \xi + \alpha_{n-1})\xi + \alpha_{n-2})\xi + \alpha_{n-3})\xi + \cdots \alpha_1)\xi + \alpha_0$$

- Using Horner's rule, it takes $O(n)$ operations in R to compute $a(\xi)$

Batch evaluation (at m points)

- Let $a = \sum_i \alpha_i x^i \in R[x]$ and $\xi_1, \xi_2, \ldots, \xi_m \in R$ be given as input with $\deg a = n$
- We want to compute $a(\xi_1), a(\xi_2), \ldots, a(\xi_m) \in R$
- Repeated application of Horner's rule achieves this in $O(mn)$ operations in R
- ... but could we do better yet again? ...

Interpolation

- Let F be a field
- Let distinct $\xi_0, \xi_1, \ldots, \xi_n \in F$ and $\eta_0, \eta_1, \ldots, \eta_n \in F$ be given as input
- We want to compute the unique polynomial $f \in F[x]$ of degree at most n that satisfies

$$f(\xi_0) = \eta_0, \quad f(\xi_1) = \eta_1, \quad \ldots, \quad f(\xi_n) = \eta_n$$

- A classical algorithm (with complexity bounded by a polynomial in n) for this task will be studied in the exercises
- ... but could we do better yet again? ...

Integral domain

- An element $a \in R$ in a ring R is a **zero divisor** if there exists a nonzero $b \in R$ with $ab = 0$
- A ring D is an **integral domain** if there are no nonzero zero divisors
- *Examples*: (of integral domains)
  $\mathbb{Z}$, any field (exercise: units are not zero divisors), $F[x]$ for a field F
- *Work point*:
  Using (1), (2), and (3), determine all zero divisors in $\mathbb{Z}_5$, $\mathbb{Z}_6$, and $\mathbb{Z}_{10}$, respectively

Greatest common divisor

- Let R be a ring and let $a, b \in R$
- We say that a **divides** b and write $a|b$ if there exists a $q \in R$ with $aq = b$
- For $a, b, c \in R$ we say that c is a **greatest common divisor** (or gcd) of a and b if
  1. $c|a$ and $c|b$,
  2. for all $d \in R$ if $d|a$ and $d|b$, then $d|c$
- A greatest common divisor need not exist, and need not be unique
- In an integral domain, any two greatest common divisors are associates

Euclidean domain

- An integral domain E together with a function $d : E \to \mathbb{Z}_{\geq 0} \cup \{-\infty\}$ is a **Euclidean domain** if for all $a, b \in E$ with $b \neq 0$ there exist $q, r \in E$ with $a = qb + r$ and $d(r) < d(b)$
- We say that $q = a \operatorname{quo} b$ is a **quotient** and $r = a \operatorname{rem} b$ a **remainder** in the division of a by b
- We assume that we have available as a subroutine a **division algorithm** that for given $a, b \in E$ with $b \neq 0$ computes $q, r \in E$ with $a = qb + r$ and $d(r) < d(b)$
- *Examples:* (of Euclidean domains)
  - $\mathbb{Z}$ with $d(a) = |a| \in \mathbb{Z}_{\geq 0}$
    - Quotient and remainder can be determined with a division algorithm for integers
  - $F[x]$ for a field F with $d(a) = \deg a$
    - Quotient and remainder can be determined with a division algorithm for polynomials

Traditional Euclidean algorithm

- Let E be an Euclidean domain
- Let $f, g \in E$ be given as input
- We seek to compute a greatest common divisor of f and g
  - Since E is an integral domain, any two greatest common divisors of f and g are related to each other by multiplication with a unit
  - The Euclidean algorithm both (a) shows that greatest common divisors *exist* and (b) gives a way of *computing* a greatest common divisor by iterative remainders
- Traditional Euclidean algorithm:
  1. $r_0 \leftarrow f$, $r_1 \leftarrow g$
  2. $i \leftarrow 1$,
     while $r_i \neq 0$ **do** $r_{i+1} \leftarrow r_{i-1} \operatorname{rem} r_i$, $i \leftarrow i + 1$
  3. **return** r_{i-1} (a greatest common divisor)
- *Why does this algorithm always stop?* (Hint: $d(r_{i+1}) < d(r_i)$)

Traditional extended Euclidean algorithm

- Let $f, g \in E$ be given as input from an Euclidean domain E
- Traditional extended Euclidean algorithm:
  1. $r_0 \leftarrow f$, $s_0 \leftarrow 1$, $t_0 \leftarrow 0$,
     $r_1 \leftarrow g$, $s_1 \leftarrow 0$, $t_1 \leftarrow 1$
  2. $i \leftarrow 1$,
     while $r_i \neq 0$ **do**
     $q_i \leftarrow r_{i-1} \operatorname{quo} r_i$
     $r_{i+1} \leftarrow r_{i-1} - q_i r_i$
     $s_{i+1} \leftarrow s_{i-1} - q_i s_i$
     $t_{i+1} \leftarrow t_{i-1} - q_i t_i$
     $i \leftarrow i + 1$
  3. $\ell \leftarrow i - 1$
     return ℓ, r_i, s_i, t_i for $i = 0, 1, \ldots, \ell + 1$, and q_i for $i = 1, 2, \ldots, \ell$

Example (over $\mathbb{Z}$)

- Let $f = 1234 \in \mathbb{Z}$ and $g = 12 \in \mathbb{Z}$
- We obtain

i	r_i	s_i	t_i	q_i
0	1234	1	0	
1	12	0	1	102
2	10	1	-102	1
3	2	-1	103	5
4	0	6	-617	

- In particular $\ell = 3$ and $r_\ell = 2$ is a greatest common divisor of 1234 and 12

Example (over $\mathbb{Z}_2[x]$)

- Let $f = x^5 + x^4 + x^3 + x^2 + x + 1 \in \mathbb{Z}_2[x]$ and $g = x^5 + x^4 + 1 \in \mathbb{Z}_2[x]$
- We obtain

i	r_i	s_i	t_i	q_i
0	$x^5 + x^4 + x^3 + x^2 + x + 1$	1	0	
1	$x^5 + x^4 + 1$	0	1	1
2	$x^3 + x^2 + x$	1	1	$x^2 + 1$
3	$x^2 + x + 1$	$x^2 + 1$	x^2	x
4	0	$x^3 + x + 1$	$x^3 + 1$	

- In particular $\ell = 3$ and $r_\ell = x^2 + x + 1$ is a greatest common divisor of $x^5 + x^4 + x^3 + x^2 + x + 1$ and $x^5 + x^4 + 1$

Analysis using invariants (in this week's problem set)

- Suppose on input $f, g \in E$ we obtain the output ℓ, r_i, s_i, t_i for $i = 0, 1, \ldots, \ell + 1$, and q_i for $i = 1, 2, \ldots, \ell$
- Introduce the matrices

$$R_0 = \begin{bmatrix} s_0 & t_0 \\ s_1 & t_1 \end{bmatrix} \in E^{2\times 2}, \qquad Q_i = \begin{bmatrix} 0 & 1 \\ 1 & -q_i \end{bmatrix} \in E^{2\times 2} \quad \text{for } i = 1, 2, \ldots, \ell,$$

  and $R_i = Q_i Q_{i-1} \cdots Q_1 R_0 \in E^{2\times 2}$ for $i = 0, 1, \ldots, \ell$
- The following invariants hold for all $i = 0, 1, \ldots, \ell$:
  1. $R_i \begin{bmatrix} f \\ g \end{bmatrix} = \begin{bmatrix} r_i \\ r_{i+1} \end{bmatrix}$.
  2. $R_i = \begin{bmatrix} s_i & t_i \\ s_{i+1} & t_{i+1} \end{bmatrix}$.
  3. r_ℓ is a greatest common divisor of r_i and r_{i+1}.
  4. $s_i f + t_i g = r_i$.

Recap of key content in Lecture 1

- A boot camp of basic concepts and definitions in algebra
- Polynomials in one variable (univariate polynomials)
- Basic tasks and first algorithms for univariate polynomials
  - addition
  - multiplication
  - division (quotient and remainder)
  - evaluation
  - interpolation (exercise)
  - greatest common divisor
- Evaluation–interpolation -duality of polynomials (exercise)
- Analysis of the extended Euclidean algorithm via invariants (exercise)

What next?

- Register to the course in Oodi if you have not already done so
  (or e-mail the lecturer in case you missed the registration period)
- Problem Set 1 available in MyCourses
- Q&A session on Thursday (12–14 hall T5)
- Problem Set 1 deadline Sun 20 Jan 20:00, Finnish time
  (submit a single PDF file — submission instructions in problem sheet)

Addendum (1/2)

- To get a hands-on perspective to the concepts and algorithm designs, it is in most cases useful to do some quick-and-dirty programming using your own favorite programming language and/or computer algebra system
- E.g. the lecturer often uses the Scala programming language for drafting out concepts and designs

  `https://www.scala-lang.org`

- Here is a `git` repository that contains a quick-and-dirty, first-draft Scala implementation (with very limited documentation) of selected concepts in this lecture:

  `https://github.com/pkaski/cs-e4500-2018.git`

Addendum (2/2)

- Computer algebra systems that you may want to try out include
  - Mathematica (https://download.aalto.fi/index-en.html)
  - GAP (https://www.gap-system.org)
  - Magma (http://magma.maths.usyd.edu.au/magma/)
  - Sage (http://www.sagemath.org)
  - ...

References I

[1] M. Agrawal, N. Kayal, and N. Saxena, PRIMES is in P, *Ann. of Math. (2)* 160 (2004), 781–793.
[doi:10.4007/annals.2004.160.781].

[2] P. Austrin, P. Kaski, and K. Kubjas, Tensor network complexity of multilinear maps, in *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA* (A. Blum, Ed.). Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019, pp. 7:1–7:21.
[doi:10.4230/LIPIcs.ITCS.2019.7].

[3] R. C. Baker, G. Harman, and J. Pintz, The difference between consecutive primes. II, *Proc. London Math. Soc. (3)* 83 (2001), 532–562.
[doi:10.1112/plms/83.3.532].

References II

[4] A. Björklund and P. Kaski, How proofs are prepared at Camelot: extended abstract, in *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016* (G. Giakkoupis, Ed.). ACM, 2016, pp. 391–400.
[doi:10.1145/2933057.2933101].

[5] R. Brent and P. Zimmermann, *Modern Computer Arithmetic*, Cambrigde University Press, 2011.
[WWW].

[6] M. L. Carmosino, J. Gao, R. Impagliazzo, I. Mihajlin, R. Paturi, and S. Schneider, Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility, in *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016* (M. Sudan, Ed.). ACM, 2016, pp. 261–270.

References III

[doi:10.1145/2840728.2840746].

[7] D. A. Cox, J. Little, and D. O'Shea, *Ideals, Varieties, and Algorithms*, fourth ed., Springer, Cham, 2015.
[doi:10.1007/978-3-319-16721-3].

[8] M. Fürer, Faster integer multiplication, *SIAM J. Comput.* 39 (2009), 979–1005.
[doi:10.1137/070711761].

[9] S. Gao, A new algorithm for decoding Reed–Solomon codes, in *Communications, Information, and Network Security* (V. K. Bhargava, H. V. Poor, V. Tarokh, and S. Yoon, Eds.), Springer, 2003, pp. 55–68.

[10] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*, third ed., Cambridge University Press, Cambridge, 2013.
[doi:10.1017/CBO9781139856065].

References IV

[11] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, Delegating computation: Interactive proofs for muggles, *J. ACM* 62 (2015), 27:1–27:64. [doi:10.1145/2699436].

[12] D. Harvey, J. van der Hoeven, and G. Lecerf, Even faster integer multiplication, *J. Complexity* 36 (2016), 1–30. [doi:10.1016/j.jco.2016.03.001].

[13] N. Karimi, P. Kaski, and M. Koivisto, Error-correcting and verifiable parallel inference in graphical models, in *Proceedings of AAAI'20*, to appear.

[14] P. Kaski, Engineering a delegatable and error-tolerant algorithm for counting small subgraphs, in *Proceedings of the Twentieth Workshop on Algorithm Engineering and Experiments, ALENEX 2018, New Orleans, LA, USA, January 7-8, 2018.* (R. Pagh and S. Venkatasubramanian, Eds.). SIAM, 2018, pp. 184–198. [doi:10.1137/1.9781611975055.16].

References V

[15] D. E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, 3rd ed., Addison-Wesley, 1998.

[16] S. Lang, *Algebra*, third ed., Springer-Verlag, New York, 2002. [doi:10.1007/978-1-4613-0041-0].

[17] N. Möller, On Schönhage's algorithm and subquadratic integer GCD computation, *Math. Comp.* 77 (2008), 589–607. [doi:10.1090/S0025-5718-07-02017-0].

[18] A. Schönhage, Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2, *Acta Informat.* 7 (1976/77), 395–398. [doi:10.1007/BF00289470].

[19] A. Schönhage and V. Strassen, Schnelle Multiplikation grosser Zahlen, *Computing (Arch. Elektron. Rechnen)* 7 (1971), 281–292.

References VI

[20] A. Shamir, How to share a secret, *Comm. ACM* 22 (1979), 612–613. [doi:10.1145/359168.359176].

[21] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, SIAM, 1992. [doi:10.1137/1.9781611970999].

[22] M. Walfish and A. J. Blumberg, Verifying computations without reexecuting them, *Commun. ACM* 58 (2015), 74–84. [doi:10.1145/2641562].

[23] R. R. Williams, Strong ETH breaks with Merlin and Arthur: Short non-interactive proofs of batch evaluation, in *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan* (R. Raz, Ed.). Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, pp. 2:1–2:17. [doi:10.4230/LIPIcs.CCC.2016.2].