# 9. Factoring integers

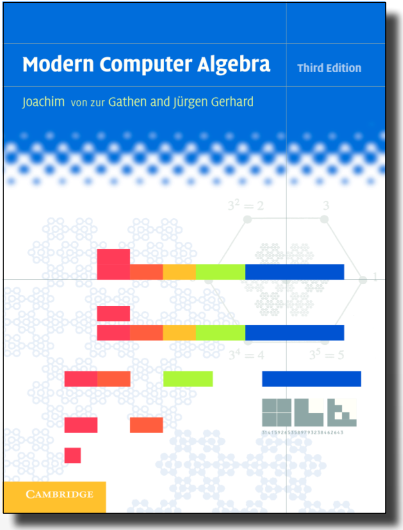
CS-E4500 Advanced Course on Algorithms
Spring 2019

**Petteri Kaski**
Department of Computer Science
Aalto University

## Motivation for this week

- A tantalizing case where the connection between polynomials and integers apparently breaks down occurs with **factoring**
- Namely, it is known how to efficiently factor a given univariate polynomial over a finite field into its irreducible components, whereas no such algorithms are known for factoring a given integer into its prime factors
- Last week we saw how to factor efficiently univariate polynomials over a finite field
- The best known algorithms for factoring integers run in time that scales moderately exponentially in the number of digits in the input; this week we study one such algorithm

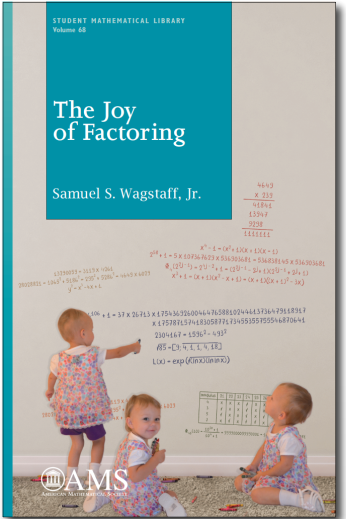
# Factoring integers




(von zur Gathen and Gerhard [11], Sections 19.1–3, 19.5)

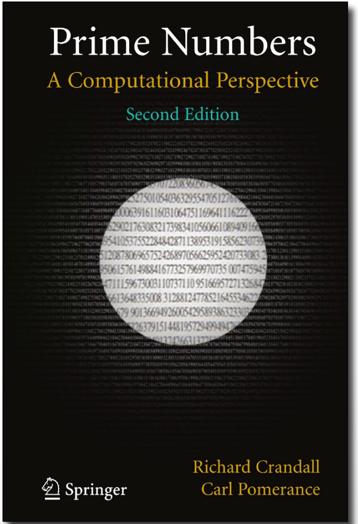
# Factoring integers

(Wagstaff [28])



# Factoring integers

(Crandall and Pomerance [7])



# Key content for Lecture 9

- **Prime numbers**, **factorization**, and **smooth numbers**
- The **prime number theorem**
- Factoring by **trial division**
- **Difference of two squares** and factoring
- Quadratic congruences, square roots (exercise), and factoring
- **Dixon's random squares algorithm** [8]

# Prime numbers

- An integer $p \in \mathbb{Z}_{\geq 2}$ is **prime** if the only positive integers that divide $p$ are $1$ and $p$
- The set $\mathbb{P} = \{2, 3, 5, 7, 11, \ldots\}$ of prime numbers is infinite
- Indeed, suppose that $p_1, p_2, \ldots, p_h$ are the $h$ least distinct primes
- Then, $p_1 p_2 \cdots p_h + 1$ is not divisible by any of the $p_1, p_2, \ldots, p_h$ and thus must have a prime divisor $p$ with $p > p_1, p_2, \ldots, p_h$

# The prime number theorem

- For $x \geq 1$, let us write $\pi(x)$ for the number of prime numbers at most $x$

Theorem 19 (Prime number theorem)

*For all* $x \geq 59$ *it holds that*

$$\frac{x}{\ln x}\left(1 + \frac{1}{2\ln x}\right) < \pi(x) < \frac{x}{\ln x}\left(1 + \frac{3}{2\ln x}\right)$$

*Proof.*

See e.g. Rosser and Schoenfeld [22, Theorem 1] □

# Factorization of an integer

- Let $N \in \mathbb{Z}_{\geq 2}$
- The **factorization** of $N$ consists of distinct primes $p_1, p_2, \ldots, p_r$ and positive integers $a_1, a_2, \ldots, a_r$ such that
$$N = p_1^{a_1} p_2^{a_2} \cdots p_r^{a_r}$$
- The primes $p_1, p_2, \ldots, p_r$ are the **prime factors** of $N$
- The factorization of $N$ is unique up to ordering of the prime factors
- We say that $N$ is a **prime power** if $r = 1$
- We say that $N$ is **squarefree** if $a_1 = a_2 = \cdots a_r = 1$

## Example: Factorization

- The factorization of $2027651281$ is

$$2027651281 = 44021 \cdot 46061$$

# Smooth integer

- Let $B \geq 2$
- Let $N \in \mathbb{Z}_{\geq 2}$ have factorization

$$N = p_1^{a_1} p_2^{a_2} \cdots p_r^{a_r}$$

- We say that $N \in \mathbb{Z}_{\geq 1}$ is $B$-**smooth** if $N = 1$ or $p_1, p_2, \ldots, p_r \leq B$

# Example: Smooth integer

- The integer $1218719480020992$ is $3$-smooth
- Indeed, the factorization of $1218719480020992$ is

$$1218719480020992 = 2^{20} \cdot 3^{19}$$

# Factoring an integer

- The **factoring problem** asks us to compute the factorization
$$N = p_1^{a_1} p_2^{a_2} \cdots p_r^{a_r}$$
for an integer $N \in \mathbb{Z}_{\geq 2}$ given as input
- To solve the factoring problem it suffices to either (i) present a **proper divisor** $d$ of $N$ with $2 \leq d \leq N - 1$, or (ii) assert that $N$ is prime
- Indeed, in case (i) we obtain the factorization of $N$ by merging the recursive factorizations of $d$ and $N/d$
- We have that $d$ is a proper divisor of $N$ if and only if $N/d$ is a proper divisor of $N$; thus, without loss of generality we can assume that a proper divisor satisfies $2 \leq d \leq \sqrt{N}$

# Trial division

- Let $N \in \mathbb{Z}_{\geq 2}$ be given as input

1. For all $d = 2, 3, \ldots, \lfloor \sqrt{N} \rfloor$
   a. If $d$ divides $N$, then output $d$ and stop
2. Assert that $N$ is prime and stop

- This algorithm runs in time $O(N^{1/2}(\log N)^c)$ for a constant $c > 0$
- We leave as an exercise the design of an algorithm that computes $\lfloor \sqrt{N} \rfloor$ in time $O((\log N)^c)$ given $N$ as input

# Detecting and factoring prime powers

- Let $N \in \mathbb{Z}_{\geq 2}$ be given as input
- In time $O((\log N)^c)$ for a constant $c > 0$ we can either output a prime $p$ and a positive integer $a$ such that $N = p^a$ or assert that $N$ is not a prime power (exercise)
- This design makes use that we can test primality in time polynomial in $\log N$ [1]

# Difference of two squares and factoring

- Suppose that $N \in \mathbb{Z}_{\geq 2}$ is odd and not a prime power
- Thus, there exist distinct odd $a, b \in \mathbb{Z}_{\geq 3}$ with

$$N = ab = \left(\frac{a+b}{2}\right)^2 - \left(\frac{a-b}{2}\right)^2$$

- Similarly, for integers $s, t \in \mathbb{Z}_{\geq 1}$ we have that

$$N = s^2 - t^2$$

implies the factorization

$$N = (s+t)(s-t)$$

# Example: Difference of two squares and factoring

- We have

$$2027651281 = 45041^2 - 1020^2$$

- Thus,

$$2027651281 = (45041 - 1020)(45041 + 1020) = 44021 \cdot 46061$$

# Quadratic congruences and square roots

- Suppose that $N \in \mathbb{Z}_{\geq 15}$ is odd and has $r \geq 2$ distinct prime factors
- Let $s^2 \equiv t^2 \pmod{N}$ with $s, t \in \{1, 2, \ldots, N-1\}$ and $\gcd(s, N) = \gcd(t, N) = 1$
- Then, there are exactly $2^r$ choices for $s$ such that $s^2 \equiv t^2 \pmod{N}$ (exercise)

# Quadratic congruences and factoring

- Suppose that $N \in \mathbb{Z}_{\geq 2}$ is odd and has $r \geq 2$ distinct prime factors
- Let $s^2 \equiv t^2 \pmod{N}$ with $s, t \in \{1, 2, \ldots, N-1\}$ and $\gcd(s, N) = \gcd(t, N) = 1$
- That is, there exists an integer $q$ with $s^2 - t^2 = (s-t)(s+t) = qN$
- We thus have that $\gcd(s+t, N)$ is a proper divisor of $N$ unless $N$ divides $s-t$ or $N$ divides $s+t$
- That is, $\gcd(s+t, N)$ is a proper divisor of $N$ unless $s \equiv \pm t \pmod{N}$
- Thus, there are $2^r - 2$ choices for $s$ such that $\gcd(s+t, N)$ is a proper divisor of $N$

# Dixon's random squares algorithm

- Let us describe Dixon's [8] random squares method of factoring
- Suppose that $N \in \mathbb{Z}_{\geq 15}$ is odd and not a prime power (in particular, $N$ has $r \geq 2$ distinct prime factors)
- We may furthermore assume that $\sqrt{N}$ is not an integer (otherwise we would have a proper divisor of $N$; computing $\lfloor \sqrt{N} \rfloor$ is an exercise)
- Let $B$ be a parameter whose value is fixed later
- The algorithm consists of three parts
  i. Find the $h = \pi(B)$ least primes $p_1 < p_2 < \cdots < p_h$ with $p_h \leq B$; if $p_j$ divides $N$ for some $j = 1, 2, \ldots, h$, then output $p_j$ and stop
  ii. Find $h + 1$ integers $2 \leq s \leq N - 2$ coprime to $N$ whose square $s^2$ rem $N$ is $B$-smooth
  iii. Find a quadratic congruence modulo $N$ using the $h + 1$ discovered integers

# Finding smooth random squares

1. Set $j \leftarrow 1$
2. While $j \leq h + 1$ do
   a. Select a uniform random $s_j \in \{2, 3, \ldots, N - 2\}$
   b. If $\gcd(s_j, N) \neq 1$ then output $\gcd(s_j, N)$ and stop
   c. Set $u \leftarrow s_j^2 \text{ rem } N$
   d. For $i = 1, 2, \ldots, h$
      i. Set $a_{ij} \leftarrow 0$
      ii. While $p_i$ divides $u$, set $a_{ij} \leftarrow a_{ij} + 1$ and $u \leftarrow u/p_i$
   e. If $u = 1$ then set $j \leftarrow j + 1$

# Example: Finding smooth random squares (1/3)

- Let $N = 2028455971$; we observe that $N$ is odd and not a prime power
- Let us work with $B = 50$ and $h = 15$ with $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_4 = 7$, $p_5 = 11$, $p_6 = 13$, $p_7 = 17$, $p_8 = 19$, $p_9 = 23$, $p_{10} = 29$, $p_{11} = 31$, $p_{12} = 37$, $p_{13} = 41$, $p_{14} = 43$, $p_{15} = 47$

# Example: Finding smooth random squares (2/3)

- Suppose we obtain the $h + 1 = 16$ smooth squares

$$
\begin{aligned}
145085^2 \text{ rem } N &= 3^3 \cdot 5 \cdot 7^3 \cdot 13 \cdot 31 \cdot 41\\
149391^2 \text{ rem } N &= 2^5 \cdot 5^2 \cdot 11 \cdot 23^2\\
154209^2 \text{ rem } N &= 2^6 \cdot 5^5 \cdot 11 \cdot 23 \cdot 29\\
159846^2 \text{ rem } N &= 2^8 \cdot 3 \cdot 7 \cdot 11^3 \cdot 13^2\\
160474^2 \text{ rem } N &= 2^{15} \cdot 7 \cdot 11 \cdot 13 \cdot 43\\
170440^2 \text{ rem } N &= 2 \cdot 13 \cdot 29^2 \cdot 31^3\\
171122^2 \text{ rem } N &= 2 \cdot 5 \cdot 7 \cdot 13 \cdot 23 \cdot 29 \cdot 31 \cdot 47\\
180169^2 \text{ rem } N &= 3^2 \cdot 5^2 \cdot 17 \cdot 31 \cdot 47\\
180200^2 \text{ rem } N &= 2^4 \cdot 3^2 \cdot 11^2 \cdot 31^2\\
180244^2 \text{ rem } N &= 2^5 \cdot 3 \cdot 5^3 \cdot 11 \cdot 13 \cdot 19\\
180376^2 \text{ rem } N &= 2^4 \cdot 3^2 \cdot 5 \cdot 11 \cdot 13 \cdot 19 \cdot 41\\
180556^2 \text{ rem } N &= 2^5 \cdot 3^3 \cdot 5^2 \cdot 11 \cdot 13 \cdot 47\\
181136^2 \text{ rem } N &= 2^4 \cdot 3^5 \cdot 5 \cdot 19 \cdot 31^2\\
181156^2 \text{ rem } N &= 2^5 \cdot 3 \cdot 5^2 \cdot 13^2 \cdot 19 \cdot 47\\
181663^2 \text{ rem } N &= 3^6 \cdot 11 \cdot 13^3 \cdot 31\\
181744^2 \text{ rem } N &= 2^4 \cdot 3^4 \cdot 5^3 \cdot 11 \cdot 17 \cdot 19
\end{aligned}
$$

# Example: Finding smooth random squares (3/3)

- We thus have the $h \times (h+1) = 15 \times 16$ matrix

$$A = \begin{bmatrix}
0 & 5 & 6 & 8 & 15 & 1 & 1 & 0 & 4 & 5 & 4 & 5 & 4 & 5 & 0 & 4 \\
3 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 2 & 1 & 2 & 3 & 5 & 1 & 6 & 4 \\
1 & 2 & 5 & 0 & 0 & 0 & 1 & 2 & 0 & 3 & 1 & 2 & 1 & 2 & 0 & 3 \\
3 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 3 & 1 & 0 & 0 & 0 & 2 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 2 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 2 & 3 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 2 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 3 & 1 & 1 & 2 & 0 & 0 & 0 & 2 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0
\end{bmatrix}$$

# A lower bound for the number of smooth squares

Lemma 20 (A lower bound for the number of smooth squares)

*Let* $S = \{s \in \mathbb{Z}_N^\times : s^2 \text{ rem } N \textit{ is } p_h\textit{-smooth}\}$ *and let* $d$ *be a positive integer with* $p_h^{2d} \leq N$*.*
*Furthermore, suppose that none of* $p_1, p_2, \ldots, p_h$ *divides* $N$*.*
*Then,*

$$|S| \geq \frac{h^{2d}}{(2d)!}$$

# Proof I

- Let us recall that we write $\mathbb{Z}_N^\times$ for the set of integers $1 \leq a \leq N-1$ coprime to $N$
- Let $Q = \{a \in \mathbb{Z}_N^\times : \exists b \in \mathbb{Z}_N^\times \;\; b^2 \equiv a \pmod N\}$
- For $x \geq 1$ and an integer $k \in \mathbb{Z}_{\geq 0}$ let us write $T_d(x)$ for the set of all integers $a \in \mathbb{Z}_{\geq 1}$ such that $a \leq x$ and there exist integers $k_1, k_2, \ldots, k_h \in \mathbb{Z}_{\geq 0}$ with $k_1 + k_2 + \ldots + k_h = k$ and $a = p_1^{k_1} p_2^{k_2} \cdots p_h^{k_h}$
- Since none of $p_1, p_2, \ldots, p_h$ divides $N$, for all $a \in T_k(x)$ we have that $a$ and $N$ are coprime
- Let $N = q_1^{e_1} q_2^{e_2} \cdots q_r^{e_r}$ be the factorization of $N$

# Proof II

- By the Chinese Remainder Theorem, we have the isomorphism

$$\mathbb{Z}_N^\times \to \mathbb{Z}_{q_1^{e_1}}^\times \times \mathbb{Z}_{q_2^{e_2}}^\times \times \cdots \times \mathbb{Z}_{q_r^{e_r}}^\times$$

  given by

$$a \mapsto (a \operatorname{rem} q_1^{e_1}, a \operatorname{rem} q_2^{e_2}, \ldots, a \operatorname{rem} q_r^{e_r})$$

- For $i = 1, 2, \ldots, r$ and $a \in \mathbb{Z}_N^\times$, let us define

$$\chi_i(a) = \begin{cases} 1 & \text{if there exists } b \in \mathbb{Z}_{q_i^{e_i}} \text{ with } b^2 \equiv a \pmod{q_i^{e_i}}, \\ -1 & \text{otherwise} \end{cases}$$

- The map $\chi(a) = (\chi_1(a), \chi_2(a), \ldots, \chi_r(a)) \in \{-1, 1\}^r$ is a homomorphism from $\mathbb{Z}_N^\times$ to $\{-1, 1\}^r$
- In particular, for all $a \in \mathbb{Z}_N^\times$ we have $a \in Q$ if and only if $\chi(a) = (1, 1, \ldots, 1)$

# Proof III

- For each $s \in \{-1, 1\}^r$, let
$$U_s = \{a \in T_d(\sqrt{N}) : \chi(a) = s\}$$
- Since $\sqrt{N}$ is not an integer, for $b, c \in U_s$ we observe that $1 \leq bc \leq N - 1$ and $\chi(bc) = (1, 1, \ldots, 1)$; in particular, $bc \in Q \cap T_{2d}(N)$
- Let $m : \cup_{s \in \{-1,1\}^r} U_s \times U_s \to V$ be a surjective map given by $(b, c) \mapsto m(b, c) = bc$
- We have $V \subseteq Q \cap T_{2d}(N)$ and $|V|\binom{2d}{d} \geq \sum_{s \in \{-1,1\}^r} |U_s|^2$
- Since every $a \in Q$ has exactly $2^r$ square roots in $\mathbb{Z}_N^\times$ (exercise), from $V \subseteq Q \cap T_{2d}(N)$ we have that $|S| \geq 2^r |V|$

## Proof IV

- Combining inequalities, we have

$$|S| \geq \binom{2d}{d}^{-1} 2^r \sum_{s \in \{-1,1\}^r} |U_s|^2$$

- By the Cauchy–Schwartz inequality and the definition of the sets $U_s$, we have

$$2^r \sum_{s \in \{-1,1\}^r} |U_s|^2 \geq \left( \sum_{s \in \{-1,1\}^r} |U_s| \right)^2 = |T_d(\sqrt{N})|^2$$

- Since $p_h^d \leq \sqrt{N}$, an element of $T_d(\sqrt{N})$ chooses exactly $d$ primes up to $p_h$, possibly with repetition; thus,

$$|T_d(\sqrt{N})| = \binom{d+h-1}{h-1} = \binom{d+h-1}{d} \geq \frac{h^d}{d!}$$

# Proof V

- Combining inequalities, we obtain

$$|S| \geq \binom{2d}{d}^{-1} \left(\frac{h^d}{d!}\right)^2 = \frac{h^{2d}}{(2d)!}$$

- This completes the proof

# Expected number of iterations

- From Lemma 20 we thus have that a uniform random $s_j \in \{2, 3, \ldots, N-2\}$ satisfies that $s_j^2 \text{ rem } N$ is $B$-smooth and $\gcd(s_j, N) = 1$ with probability at least $\frac{|S|-2}{N-3} \geq \frac{\frac{h^{2d}}{(2d)!}-2}{N} \geq \frac{h^{2d}}{2(2d)!N}$ where in the last inequality we have assumed that $\frac{h^{2d}}{(2d)!} \geq 4$; here $h = \pi(B)$ and $d$ is a positive integer with $p_h^{2d} \leq N$
- Thus, in expectation we need at most $2(h+1)\frac{(2d)!N}{h^{2d}}$ iterations of the while loop to find $h+1$ smooth random squares
- Let $B = N^{\frac{1}{2d}}$ and recall from Theorem 19 that for all large enough $B$ we have both $h = \pi(B) > \frac{B}{\ln B}$ and $2(h+1) \leq B$
- Since $(2d)! \leq (2d)^{2d}$, in expectation the number of iterations is at most

$$(h+1)\frac{(2d)!N}{h^{2d}} < B\frac{(2d\ln B)^{2d}N}{B^{2d}} = N^{1/2d}(\ln N)^{2d}$$

# Finding a quadratic congruence modulo $N$ (1/2)

- Let us now turn to the last part of the algorithm that finds a quadratic congruence modulo $N$
- The coefficients $a_{ij}$ for $i = 1, 2, \ldots, h$ and $j = 1, 2, \ldots, h + 1$ from an $h \times (h + 1)$ integer matrix
- The $h + 1$ columns of this matrix are linearly dependent modulo $2$

1. Find $\epsilon_j \in \{0, 1\}$ for $j = 0, 1, \ldots, h + 1$ such that $\epsilon_j = 1$ for at least one $j$ and, for all $i = 1, 2, \ldots, h$, we have

$$\sum_{j=1}^{h+1} a_{ij}\epsilon_j \equiv 0 \pmod{2} \tag{36}$$

- Since $h \leq B$, this can be done in time $O(B^3)$ using, for example, Gaussian elimination
- Let $\ell = 1, 2, \ldots, h + 1$ with $\epsilon_\ell = 1$ and $\epsilon_j = 0$ for all $j = \ell + 1, \ell + 2, \ldots, h + 1$

# Finding a quadratic congruence modulo $N$ (2/2)

2. Next, set

$$s \leftarrow s_1^{\epsilon_1} s_2^{\epsilon_2} \cdots s_\ell^{\epsilon_\ell} \text{ rem } N$$

3. For all $i = 1, 2, \ldots, h$, set

$$d_i = \frac{1}{2} \sum_{j=1}^{\ell} a_{ij} \epsilon_j$$

   and observe from (36) that $d_i$ is a nonnegative integer

4. Set

$$t \leftarrow p_1^{d_1} p_2^{d_2} \cdots p_h^{d_h} \text{ rem } N$$

- By construction we now have $s^2 \equiv t^2 \pmod{N}$

# Example: Finding a quadratic congruence modulo $N$ (1/2)

- Let us continue working with $N = 2028455971$;
  recall the smooth squares and the $h \times (h+1)$ matrix $A$ from the earlier example
- We have $A\epsilon \equiv 0 \pmod{2}$ for the vector $\epsilon \in \{0,1\}^{h+1}$ with

$$\epsilon = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^\top$$

- We also have $d = (A\epsilon)/2$ with

$$d = \begin{bmatrix} 12 & 4 & 7 & 1 & 3 & 2 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}^\top$$

- Accordingly,

$$s = (154209 \cdot 159846 \cdot 171122 \cdot 180169 \cdot 180244 \cdot 181744) \operatorname{rem} N = 1840185960$$

  and

$$t = (2^{12} \cdot 3^4 \cdot 5^7 \cdot 7 \cdot 11^3 \cdot 13^2 \cdot 17 \cdot 19 \cdot 23 \cdot 29 \cdot 31 \cdot 47) \operatorname{rem} N = 1325950600$$

# Example: Finding a quadratic congruence modulo $N$ (2/2)

- We have

$$\begin{aligned} N &= 2028455971 \\ s &= 1840185960 \\ t &= 1325950600 \end{aligned}$$

- We readily check that $s^2 \equiv t^2 \pmod{N}$
- Furthermore, we have

$$\begin{aligned} \gcd(s+t, N) &= 46073 \\ \gcd(s-t, N) &= 44027 \end{aligned}$$

  which splits $N = 46073 \cdot 44027 = 2028455971$ into two proper divisors (which are in fact prime)

# The probability to obtain a proper divisor of $N$

- We claim that with probability at least $1/2$ it holds that $s \not\equiv \pm t \pmod{N}$ and thus $\gcd(s+t, N)$ is a proper divisor of $N$
- Indeed, observe that the coefficients $a_{ij}$ depend only on the values $s_j^2 \text{ rem } N$; thus, $t$ depends only on the values $s_j^2 \text{ rem } N$
- Condition on the values $s_1, s_2, \ldots, s_{\ell-1}$ and study the distribution of the value $s_\ell$ conditioned on the value $s_\ell^2 \text{ rem } N$
- We have that $s_\ell$ has $2^r$ possible values (the $2^r$ possible square roots of $s_\ell^2 \text{ rem } N$), exactly $2$ of which lead to the outcome $s \equiv t \pmod{N}$ or $s \equiv -t \pmod{N}$ since $\epsilon_\ell = 1$
- Because each possible value of $s_\ell$ occurs with probability $2^{-r}$, in aggregate (over all conditionings) we have that $s \equiv t \pmod{N}$ or $s \equiv -t \pmod{N}$ with probability at most $2^{1-r} \leq 1/2$ since $r \geq 2$

# Expected running time (1/2)

- Recall that we let $B = N^{\frac{1}{2d}}$ for a positive integer $d = d(N)$ such that $B \to \infty$ and $(B/(2d \ln B))^{2d} \geq 4$ as $N \to \infty$
- Recall that we obtain $h + 1$ smooth squares in expectation in at most $N^{1/2d}(\ln N)^{2d}$ iterations of the while loop
- Since $h \leq B$, each iteration runs in time $O(B(\log N)^c)$ for a constant $c > 0$
- The time to find a quadratic congruence modulo $N$ is at most $O(B^3 + B(\log N)^c)$ for a constant $c > 0$
- Thus, the entire algorithm runs in at most $O(N^{\frac{3}{2d}}(\ln N)^{2d}(\log N)^c)$ expected time and outputs a proper divisor of $N$ with probability at least $1/2$

## Expected running time (2/2)

- Recall that the expected running time is at most $O(N^{\frac{3}{2d}}(\ln N)^{2d}(\log N)^c)$
- Observe that $N^{\frac{3}{2d}}(\ln N)^{2d} = \exp(\frac{3\ln N}{2d} + 2d\ln\ln N)$
- Solve $\frac{\ln N}{d} = d\ln\ln N$ and round up to obtain

$$d = \left\lceil \sqrt{\frac{\ln N}{\ln\ln N}} \right\rceil$$

  and thus, for all large enough $N$,

$$N^{\frac{3}{2d}}(\ln N)^{2d}(\log N)^c = \exp(O(\sqrt{\ln N \ln\ln N}))$$

- Since $B = N^{1/(2d)} \geq \exp(\sqrt{\ln N})$ for all large enough $N$, we obtain expected running time at most $\exp(O(\sqrt{\ln N \ln\ln N}))$ for Dixon's algorithm

## Remarks

- We have here barely scratched the surface of moderately-exponential-time randomized algorithms for factoring integers
- To obtain a *practical* algorithm design that runs in moderately exponential time (based on a heuristic analysis), more work is needed—the aforementioned exposition and analysis of Dixon's algorithm merely illustrates some of the key theoretical ideas
- Cf. Crandall and Pomerance [7] and Wagstaff [28] for a more comprehensive introduction to integer factoring algorithms

# Key content for Lecture 9

- **Prime numbers**, **factorization**, and **smooth numbers**
- The **prime number theorem**
- De Bruijn's lower bound for smooth numbers (exercise)
- Factoring by **trial division**
- The factorial function and factoring—fast polynomial evaluation and the **Pollard–Strassen algorithm** [21, 26]
- **Difference of two squares** and factoring
- Quadratic congruences, square roots (exercise), and factoring
- **Dixon's random squares algorithm** [8]

# Lecture schedule

| | |
|---|---|
| Tue 15 Jan: | 1. Polynomials and integers |
| Tue 22 Jan: | 2. The fast Fourier transform and fast multiplication |
| Tue 29 Jan: | 3. Quotient and remainder |
| Tue 5 Feb: | 4. Batch evaluation and interpolation |
| Tue 12 Feb: | 5. Extended Euclidean algorithm and interpolation from erroneous data |
| *Tue 19 Feb:* | *Exam week — no lecture* |
| Tue 27 Feb: | 6. Identity testing and probabilistically checkable proofs |
| *Tue 5 Mar:* | *Break — no lecture* |
| Tue 12 Mar: | 7. Finite fields |
| Tue 19 Mar: | 8. Factoring polynomials over finite fields |
| Tue 26 Mar: | 9. Factoring integers |

# Learning objectives (1/2)

- Terminology and objectives of modern algorithmics, including elements of algebraic, online, and randomised algorithms
- Ways of coping with uncertainty in computation, including error-correction and proofs of correctness
- The art of solving a large problem by reduction to one or more smaller instances of the same or a related problem
- (Linear) independence, dependence, and their abstractions as enablers of efficient algorithms

# Learning objectives (2/2)

- Making use of duality
  - Often a problem has a corresponding **dual** problem that is obtainable from the original (the **primal**) problem by means of an easy transformation
  - The primal and dual control each other, enabling an algorithm designer to use the interplay between the two representations
- Relaxation and tradeoffs between objectives and resources as design tools
  - Instead of computing the exact optimum solution at considerable cost, often a less costly but principled approximation suffices
  - Instead of the complete dual, often only a randomly chosen partial dual or other relaxation suffices to arrive at a solution with high probability

# CS-E4500 Advanced Course in Algorithms (5 ECTS, III–IV, Spring 2019)

| 2019 | KALENTERI | | | | 2019 |
|---|---|---|---|---|---|
| **Tammikuu** | **Helmikuu** | **Maaliskuu** | **Huhtikuu** | **Toukokuu** | **Kesäkuu** |
| 1 Ti Uudenvuodenpäivä | 1 Pe | 1 Pe | 1 Ma Vk 14 T9 | 1 Ke Vappu | 1 La |
| 2 Ke | 2 La | 2 La | 2 Ti | 2 To | 2 Su |
| 3 To | 3 Su D3 | 3 Su | 3 Ke | 3 Pe | 3 Ma Vk 23 |
| 4 Pe | 4 Ma Vk 06 T3 | 4 Ma Vk 10 | 4 To | 4 La | 4 Ti |
| 5 La | 5 Ti L4 | 5 Ti Laskiainen | 5 Pe | 5 Su | 5 Ke |
| 6 Su Loppiainen | 6 Ke | 6 Ke Break | 6 La | 6 Ma Vk 19 | 6 To |
| 7 Ma Vk 02 | 7 To Q4 | 7 To | 7 Su | 7 Ti | 7 Pe |
| 8 Ti | 8 Pe | 8 Pe | 8 Ma Vk 15 | 8 Ke | 8 La |
| 9 Ke | 9 La | 9 La | 9 Ti | 9 To | 9 Su Helluntaipäivä |
| 10 To | 10 Su D4 | 10 Su D6 | 10 Ke | 10 Pe | 10 Ma Vk 24 |
| 11 Pe | 11 Ma Vk 07 T4 | 11 Ma Vk 11 T6 | 11 To | 11 La | 11 Ti |
| 12 La | 12 Ti L5 | 12 Ti L7 | 12 Pe | 12 Su Äitienpäivä | 12 Ke |
| 13 Su | 13 Ke | 13 Ke | 13 La | 13 Ma Vk 20 | 13 To |
| 14 Ma Vk 03 | 14 To Q5 | 14 To Q7 | 14 Su Palmusunnuntai | 14 Ti | 14 Pe |
| 15 Ti L1 | 15 Pe | 15 Pe | 15 Ma Vk 16 | 15 Ke | 15 La |
| 16 Ke | 16 La | 16 La | 16 Ti | 16 To | 16 Su |
| 17 To Q1 | 17 Su | 17 Su D7 | 17 Ke | 17 Pe | 17 Ma Vk 25 |
| 18 Pe | 18 Ma Vk 08 | 18 Ma Vk 12 T7 | 18 To | 18 La | 18 Ti |
| 19 La | 19 Ti Exam | 19 Ti L8 | 19 Pe Pitkäperjantai | 19 Su Kaatuneiden muistopäivä | 19 Ke |
| 20 Su D1 | 20 Ke | 20 Ke Kevätpäiväntasaus | 20 La | 20 Ma Vk 21 | 20 To |
| 21 Ma Vk 04 T1 | 21 To week | 21 To Q8 | 21 Su Pääsiäispäivä | 21 Ti | 21 Pe Kesäpäivänseisaus |
| 22 Ti L2 | 22 Pe | 22 Pe | 22 Ma 2. pääsiäispäivä | 22 Ke | 22 La Juhannus |
| 23 Ke | 23 La | 23 La | 23 Ti | 23 To | 23 Su |
| 24 To Q2 | 24 Su D5 | 24 Su D8 | 24 Ke | 24 Pe | 24 Ma Vk 26 |
| 25 Pe | 25 Ma Vk 09 T5 | 25 Ma Vk 13 T8 | 25 To | 25 La | 25 Ti |
| 26 La | 26 Ti L6 | 26 Ti L9 | 26 Pe | 26 Su | 26 Ke |
| 27 Su D2 | 27 Ke | 27 Ke | 27 La | 27 Ma Vk 22 | 27 To |
| 28 Ma Vk 05 T2 | 28 To Q6 | 28 To Q9 | 28 Su | 28 Ti | 28 Pe |
| 29 Ti L3 | | 29 Pe | 29 Ma Vk 18 | 29 Ke | 29 La |
| 30 Ke | | 30 La | 30 Ti | 30 To Helatorstai | 30 Su |
| 31 To Q3 | | 31 Su Kesäaika alkaa D9 | | 31 Pe | |

L = Lecture; hall T5, Tue 12–14

Q = Q & A session; hall T5, Thu 12–14

D = Problem set deadline; Sun 20:00

T = Tutorial (model solutions); hall T6, Mon 16–18

# 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] 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].

[3] 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].

# References II


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

[5] 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.
[doi:10.1145/2840728.2840746].

[6] 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].

# References III


[7] R. Crandall and C. Pomerance, *Prime Numbers: A Computational Perspective*, second ed., Springer, New York, 2005.
[doi:10.1007/0-387-28979-8].

[8] J. D. Dixon, Asymptotically fast factorization of integers, *Math. Comp.* 36 (1981), 255–260.
[doi:10.2307/2007743].

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

[10] 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.

# References IV


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

[12] 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].

[13] 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].


[14] 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].

# References V


[15] 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].

[16] K. S. Kedlaya and C. Umans, Fast polynomial factorization and modular composition, *SIAM J. Comput.* 40 (2011), 1767–1802. [doi:10.1137/08073408X].

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

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

# References VI

[19] R. Lidl and H. Niederreiter, *Finite fields*, second ed., Cambridge University Press, Cambridge, 1997.

[20] 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].

[21] J. M. Pollard, Theorems on factorization and primality testing, *Proc. Cambridge Philos. Soc.* 76 (1974), 521–528.

[22] J. B. Rosser and L. Schoenfeld, Approximate formulas for some functions of prime numbers, *Ill. J. Math.* 6 (1962), 64–94.
[WWW].

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

# References VII


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

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

[26] V. Strassen, Einige Resultate über Berechnungskomplexität, *Jber. Deutsch. Math.-Verein.* 78 (1976/77), 1–8.

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

[28] S. S. Wagstaff, Jr., *The Joy of Factoring*, American Mathematical Society, Providence, RI, 2013. [doi:10.1090/stml/068].

# References VIII


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

[30] 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].