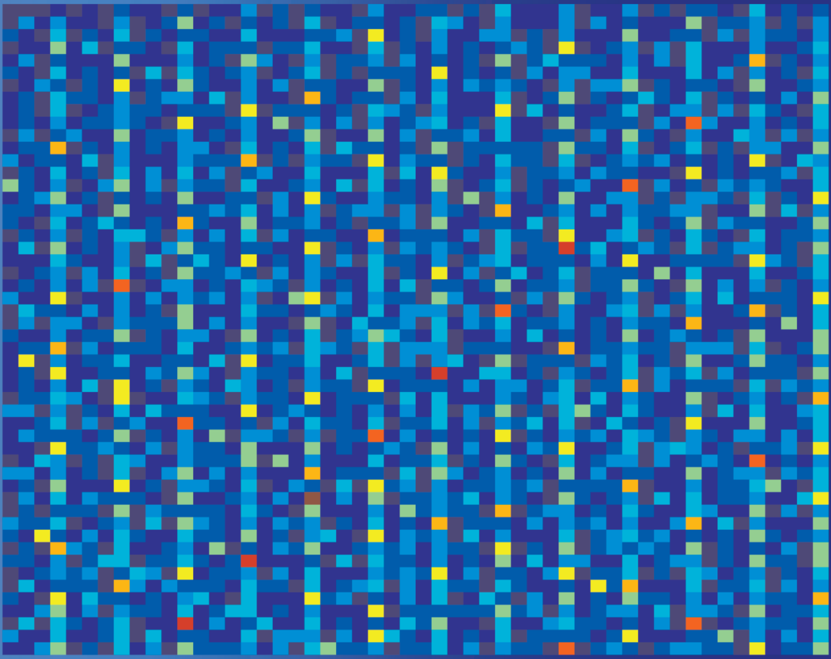
STUDENT MATHEMATICAL LIBRARY
Volume 70

Primality Testing for Beginners

Lasse Rempe-Gillen
Rebecca Waldecker



AMS
American Mathematical Society

Primality Testing for Beginners

STUDENT MATHEMATICAL LIBRARY
Volume 70

Primality Testing for Beginners

Lasse Rempe-Gillen
Rebecca Waldecker

AMS
American Mathematical Society
Providence, Rhode Island

The cover illustration is a variant of the Sieve of Eratosthenes (Section 1.5), showing the integers from 1 to 2704 colored by the number of their prime factors, including repeats. The illustration was created using MATLAB. The back cover shows a phase plot of the Riemann zeta function (see Appendix A), which appears courtesy of Elias Wegert (www.visual.wegert.com).

2010 *Mathematics Subject Classification.* Primary 11-01, 11-02, 11Axx, 11Y11, 11Y16.

For additional information and updates on this book, visit
www.ams.org/bookpages/stml-70

Library of Congress Cataloging-in-Publication Data

Rempe-Gillen, Lasse, 1978– author.
[Primzahltests für Einsteiger. English]
Primality testing for beginners / Lasse Rempe-Gillen, Rebecca Waldecker.
pages cm. — (Student mathematical library ; volume 70)
Translation of: Primzahltests für Einsteiger : Zahlentheorie - Algorithmik - Kryptographie.
Includes bibliographical references and index.
ISBN 978-0-8218-9883-3 (alk. paper)
1. Number theory. I. Waldecker, Rebecca, 1979– author. II. Title.

QA241.R45813 2014
512.7′2—dc23

2013032423

∞ The paper used in this book is acid-free and falls within the guidelines
established to ensure permanence and durability.
Visit the AMS home page at http://www.ams.org/

10 9 8 7 6 5 4 3 2 1 18 17 16 15 14

For our spouses,
Emma and Lars

Contents

Part 2. The AKS Algorithm

Preface

"How on earth can you do research in mathematics?", we are often asked. The idea of ongoing research in other sciences is deeply embedded in the public consciousness, not only in those fields that may lead to self-driving cars or new life-saving drugs, but also in the most theoretical areas such as particle physics, where scientists try to puzzle out the likely nature of matter on the smallest possible scales. In contrast, even among members of the public who profess their enjoyment of mathematics there is little awareness that many mathematical questions remain open and that these are the subject of intensive investigation.

One of the difficulties in challenging this perception lies in the highly specialized nature of modern mathematics itself. Even professional mathematicians are usually unable to fully appreciate research advances outside their own branches of mathematics. Of course there are exceptions to this rule, such as **Fermat's Last Theorem**, which was proved by Andrew Wiles in the 1990s and could hardly be simpler to state: "If $n > 2$, then there are no non-zero integers a, b, and c such that $a^n + b^n = c^n$." Yet in this case the proof is particularly long and difficult, and only a small number of experts worldwide are able to fully comprehend it.

In the summer of 2002, the computer scientist Manindra Agrawal and his students Neeraj Kayal and Nitin Saxena achieved a remarkable feat: they discovered an efficient and deterministic test for the primality of a natural number. (To learn about the meaning of these notions, keep reading!) We were fascinated by this result not only because it answers a long-standing open question, but because the mathematics behind it is beautiful and, compared to other modern research advances, extremely accessible. We decided to use this opportunity to get young people in touch with actual mathematical research by offering a course on the subject at a German summer academy for secondary-school students in 2005. Inspired by the participants and their enthusiasm, we started to transform the course material into a book manuscript, the German version of which appeared in 2009. The book you hold in your hands is an English-language edition of this text. During the translation process, we have also corrected a few errors, changed some of the presentation for pedagogical reasons, and updated the content with new results, where appropriate.

We thank the students from our original summer course for their motivation and dedication and all those who have since helped us for their comments and suggestions regarding both the German and English editions. We also thank you – our reader – for your interest in this little volume, and we hope that you will enjoy reading the book as much as we did writing it.

Lasse Rempe-Gillen and Rebecca Waldecker
August 2013

Introduction

Most of us encounter *prime numbers* for the first time in secondary school: a number is *prime* if it has exactly two divisors, namely 1 and the number itself. We also learn that every natural number can be written as a product of its prime factors – for example $2013 = 3 \cdot 11 \cdot 61$ and the numbers 3, 11, and 61 are primes. However, it is rarely emphasized in classrooms that this is only the beginning of a long story, in which mathematicians working in the area of "number theory" have been discovering the secrets of prime numbers for thousands of years. Moreover, the story is far from over: many questions remain unsolved, with no solutions currently in sight. (A few open problems can be found in Appendix A.)

Also, many people are unaware that they use prime numbers almost every day. As recently as in the year 1940, the great English mathematician G. H. Hardy wrote in his book *A Mathematician's Apology* [**Har**] that number theory has no conceivable practical applications but that it deserves to be studied for its beauty alone. However, advances in information technology during the second half of the twentieth century led researchers to look for secure methods of electronic transmission of information. In the process, they proved Hardy wrong about the applicability of number theory (though not about its beauty). In 1977, the computer scientists Ronald Rivest, Adi Shamir, and Leonard Adleman developed a procedure, now known as the **RSA algorithm**, that allows the secure transmission of a message to which no one but the sender and the receiver should have access. This is the foundation of all encryption methods commonly used today, e.g. for online credit card purchases or online banking.

We will study the RSA method more closely in Section 4.2. Its basic idea is the following surprising principle:

> It is (relatively) **easy** to decide whether or not a given number is prime. However, it is **hard** to find the prime divisors of a given composite (i.e. non-prime) number.

Considering our knowledge about prime numbers from school, this is a remarkable claim. For example, we can use an ancient method known as the *Sieve of Eratosthenes* (Section 1.5) to test whether a given number is prime. If it is not, this procedure will also provide us with a list of its prime divisors.

However, numbers routinely used in encryption algorithms today may have several hundreds or even thousands of digits. Anyone who has used the Sieve of Eratosthenes by hand, e.g. to find all prime numbers less than 200 (see Exercise 1.5.1), will find it easy to believe that this method cannot be carried out in practice for numbers of that size – even using the most advanced computer technology. To encourage research in this direction, *RSA Laboratories* (a leading security company) dared experts and puzzlers worldwide from 1991 to 2007 to break the RSA encryption scheme in the **Factoring Challenge**. They published a list of so-called **RSA numbers** (products of two different, extremely large primes), daring the public to find the two prime factors. In some cases, rather large amounts of money were offered for the solution. Although the challenge was officially closed in 2007, many factorizations still remain unknown!

So it is difficult to find prime factors. But why should detecting primality be a simpler problem, as claimed above? The key is to find properties of prime numbers that do not require excluding the presence of divisors and can therefore be checked much more efficiently.

Just such a property of prime numbers was already discovered in 1640 by the French lawyer and mathematician Pierre de Fermat. In

```
RSA-2048 = 2519590847565789349402718324004839857
      14292821262040320277771378360436620207075955
      56264018525880784406918290641249515082189298
      55914917618450280848912007284499268739280728
      77767359714183472702618963750149718246911650
      77613379859095700097330459748808428401797429
      10064245869181719511874612151517265463228221
      68699875491824224336372590851418654620435767
      98423387184774447920739934236584823824281198
      16381501067481045166037730605620161967625613
      38441436038339044149526344321901146575444541
      78424020924616515723350778707749817125772467
      96292638635637328991215483143816789988504044
      53640235273819513786365643912120103971228221
      20720357
```

Until 2007, anyone who found the prime factors of the number "RSA-2048" would have received a prize of US$200,000. No factorization is known to the present day.

the 1970s, computer scientists Gary Miller and Michael Rabin refined this property to obtain a practical primality test. Their test is still used today when encrypting messages using the RSA method, demonstrating that mathematical theories developed without aspirations of real-life applications can turn out to have unexpected and extremely important practical consequences.

Curiously, the Miller-Rabin method of primality testing is **randomized**, meaning that it relies on a random choice of certain parameters. Hence there is a (small) chance that the procedure does not provide a correct answer after a reasonable amount of time. As one can ensure that the probability of error is negligible, this element of chance does not have real disadvantages in practice. However, from a theoretical perspective it is natural to ask whether randomization is really necessary: is there an efficient method for primality testing that is **deterministic**, i.e. does not require the use of random numbers?

This problem remained unsolved for decades, until the Indian computer scientists Agrawal, Kayal, and Saxena proposed an elegant solution in 2002. Due to its fundamental importance and the elementary nature of the methods employed, their result received great attention throughout mathematics. The article was immediately celebrated as a “Breakthrough for Everyone” [**Bo**] and appeared in 2004 in the *Annals of Mathematics*, one of the most prestigious mathematical journals [**AKS**].

The objective of this book is to give a complete presentation of the proof of the theorem of Agrawal, Kayal, and Saxena, without requiring any prior knowledge beyond general computational skills and the ability to think logically. As part of this presentation, we naturally develop the prerequisites from mathematics and computer science that are needed to understand the proof and to appreciate its importance. We hope that, at the same time, the reader will catch a glimpse of the beauty of mathematics and obtain an impression of how many interesting questions still remain open.

About this book. The book is aimed at interested high school pupils and teachers, but also at undergraduate students in mathematics and computer science (to whom it should be accessible from the first year). It can be used as the textbook for a summer school or a reading course.

It is not our intention to primarily give an introduction to the theory of numbers or algorithms. There are already many excellent such books – the reader will find some of them in the references at the end of the relevant chapters. On the other hand, our book is not a work of mathematical research; it is written neither by nor for experts. Research mathematicians and computer scientists might feel that the original article by Agrawal, Kayal, and Saxena or other sources (such as the book *Primality Testing in Polynomial Time* by Dietzfelbinger [**Dtz**] that is written for a more advanced audience) proceed at a more appropriate pace for them.

Instead, we shall focus on one main goal – the treatment of the algorithm of Agrawal, Kayal, and Saxena, henceforth referred to as the “AKS algorithm” – throughout the whole book. Thus we shall cover precisely those concepts that are part of the required background for this result. At the same time, we gently introduce the reader to the world of mathematical proof. As far as we know, this approach to a complete treatment of a recent mathematical breakthrough separates our text from other books written for the same audience.

The first four chapters are designed mainly to introduce the reader to number theory and algorithm theory, as far as required for the AKS algorithm. We also give a brief historical and mathematical survey of cryptography (the science of encryption). In content and scope, we stay close to the material that was covered in our course at the Deutsche SchülerAkademie.

In the second part of the book, we essentially present the content of the AKS article [**AKS**], referring to the mathematics learned in the first part and developing further "ingredients" when necessary. We take care to both explain the underlying ideas and present the proof correctly and in detail. Readers with solid background knowledge can skip the first part of the book and give the AKS algorithm a try immediately, looking back when necessary.

Numerous exercises and comments are included at the end of each section to provide further background to the reader. The purpose of the exercises is not only to confirm that the new ideas from the section have been understood, but they are also meant as a general invitation to "learning by doing". From our experience, this is the best way to learn mathematics. Also, as a reader, one might appreciate certain ideas – particularly if they seem natural in hindsight – much more after several hours or even days spent thinking about a solution! We intentionally did not order the exercises according to difficulty. Those that require the use of an electronic computer or calculator are marked with "(P)" and those that will be used later in the book with "(!)". At the end of a section, there are usually further (and possibly more difficult) exercises and comments. These are meant to invite the reader to learn more about the corresponding subject but can be omitted at the first reading. Appendix A discusses open problems regarding prime numbers, while Appendix B contains solutions or hints to all exercises marked with "(!)". Complete solutions and our contact details will be provided at the website `www.ams.org/bookpages/stml70`. We appreciate all comments, corrections of mistakes (including typographical errors), and, of course, all questions or suggestions for improvement.

Proofs. "Proofs" are a central concept in mathematics. They establish the truth of mathematical statements beyond all doubt. In a proof, we usually start with the given **hypothesis** and deduce the desired **conclusion** using a number of logical steps. Sometimes we change perspective and begin with the assumption that the conclusion does *not* hold, deriving a **contradiction** to our hypothesis or to known facts. In school, proofs often receive little attention and can seem mysterious or difficult to understand, so we shall take care to explain our arguments very carefully and in detail.

A proof can also be viewed as an **explanation**, helping the reader to understand why a claim is true. This is precisely what we aim to achieve in our book. Assuming no prior knowledge apart from elementary rules of calculation that are familiar from school, the reader will learn all necessary prerequisites to understand the work of Agrawal, Kayal, and Saxena. We attempt to point out the underlying ideas clearly and to explain the separate logical steps carefully. For this reason, we sometimes refrain from using the shortest or most elegant arguments, in the hope that our treatment can provide the reader with a deeper understanding of the material.

This book will not only familiarize the readers with the principles of mathematical proof but should also enable them to deduce simple results themselves. Thus, in later chapters, we sometimes relegate parts of the arguments to exercises and give generous hints.

Definitions, lemmas, and theorems. In mathematics, it is important that all terms be clearly defined before they are used, i.e. that their meanings be rigorously established. Such an introduction of mathematical notation or of a new concept is called a **definition**. Once a mathematical object or a mathematical property has been defined, we can investigate it and try to prove certain facts about it. We distinguish between more difficult or more significant results and those that may be of a more auxiliary nature (e.g. established on the way to the proof of a more important fact). The former are referred to as **theorems** whereas the latter will be called **lemmas**. Which of these two categories a given fact is placed in may, however, depend on personal taste! Lemmas and theorems are both formulated in the same manner, beginning with a hypothesis and ending with a conclusion that is claimed to follow from the hypothesis. Therefore all lemmas and theorems require a proof. Results that follow in a simple manner from a previously proved fact are referred to as **corollaries**.

To make references easier to find, theorems, lemmas, definitions, exercises, etc., are labelled consecutively within each section.

If a statement takes the form "A implies B", then the **converse** of this statement is "B implies A". An implication and its converse usually have very different meanings; for example, "if 6 divides n, then 3 divides n" is always true, while its converse, "if 3 divides n, then 6 divides n", is false in general. When both the implication "A implies B" and its converse hold, we also say that A holds **if and only if** B does. For example, 6 divides n if and only if n is even and 3 divides n.

Mathematical notation. In school, we encounter the following collections of numbers:

- the **natural numbers** $\mathbb{N} = \{1, 2, 3, 4, \dots\}$;
- the **integers** $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$;
- the **even integers**, i.e. the set $\{\dots, -4, -2, 0, 2, 4, \dots\}$;
- the **odd integers**, i.e. the set $\{\dots, -3, -1, 1, 3, 5, \dots\}$;
- the **rational numbers** $\mathbb{Q} = \left\{\frac{p}{q} : p, q \in \mathbb{Z}, q \neq 0\right\}$;
- the **real numbers** $\mathbb{R}$ (i.e., the full number line).

If a and b are numbers from one of these sets, then $a \leq b$ and $a < b$ stand for "a is less than or equal to b" and "a is strictly less than b", respectively. The notation $a \geq b$ and $a > b$ is defined analogously.

We explicitly note that zero is *not* a natural number according to our definition. (There is no general agreement on this among mathematicians.) Hence we also define

$$\mathbb{N}_0 := \{a \in \mathbb{Z} : a \geq 0\}. \tag{$*$}$$

Here the symbol $:=$ means "is defined as". It is used to introduce an abbreviation or a new notation (*not* to claim the equality of two previously defined quantities). So we can read $(*)$ as "$\mathbb{N}_0$ denotes the collection of all non-negative integers".

Collections as above are called **sets** in mathematics. That is, a set is a collection of distinct objects (its **elements**); two sets are equal if they have the same elements. The sets we encounter will almost exclusively consist of numbers, rather than more complicated objects. As the basic notation of set theory – as usually encountered in school – will be used routinely throughout the book, let us briefly review it here. If M is a set, then $x \in M$ means "x is an element of the set M". When y does *not* belong to M, we write $y \notin M$. A set N is called a **subset** of a set M if every element of N is also an element of M. In this case we write $N \subseteq M$. For example,

$$\mathbb{N} \subseteq \mathbb{N}_0 \subseteq \mathbb{Z} \subseteq \mathbb{Q} \subseteq \mathbb{R}.$$

Note that, by definition, every set is a subset of itself. If $N \subseteq M$ and $N \neq M$, then N is called a **proper subset** of M, and we write $N \subsetneq M$. The symbol $\varnothing$ denotes the **empty set**: the (unique) set that has no elements. The number of elements of a set M is denoted by $\#M$; for example $\#\{2,4,6,8,10\} = 5$ and $\#\mathbb{N} = \infty$. (The symbol ∞ stands, as usual, for "infinity".) As already encountered in $(*)$,

$$\{x \in M : x \text{ has the property } \dots\}$$

denotes the set of all elements of M that have the stated property.

We use the standard notation from school for addition, subtraction, multiplication, and division, as well as for powers. Occasionally we omit the dot for multiplication, writing (for example) $3x$ instead of $3 \cdot x$. Elementary rules of calculation, such as the distributive law $a(b+c) = ab + ac$, are used routinely throughout.

If $x_1, \dots, x_n$ are real numbers, with $n \in \mathbb{N}$, then their sum and product are abbreviated as follows:

$$\sum_{i=1}^{n} x_i = x_1 + x_2 + \dots + x_n; \qquad \prod_{i=1}^{n} x_i = x_1 \cdot x_2 \cdots x_n.$$

For example, $\sum\limits_{i=1}^{n} i = 1 + 2 + \dots + n$ and $\sum\limits_{i=1}^{n} \frac{1}{i} = \frac{1}{1} + \frac{1}{2} + \dots + \frac{1}{n}$.

If n is any natural number, then $n!$ denotes the **factorial** of n, defined by

$$n! := \prod_{i=1}^{n} i \quad \Big(= 1 \cdot 2 \cdots (n-1) \cdot n \Big).$$

For example, $1! = 1$, $3! = 6$, and $5! = 120$. We also define $0! := 1$.

Recall the following rules for transforming powers: if a, b, x, y are real numbers with $a, b > 0$, then

$$a^x \cdot a^y = a^{x+y}, \quad (ab)^x = a^x \cdot b^x, \quad \text{and} \quad (a^x)^y = a^{x \cdot y}.$$

Instead of $a^{(x^y)}$ we write a^{x^y}. In general, this is *not* the same as $(a^x)^y$: for example we have $2^{3^2} = 512$, but $(2^3)^2 = 64$. By definition, raising any number to the power zero yields 1: $a^0 := 1$ for all $a \in \mathbb{R}$. We say that a natural number n is a **perfect power** of $a \in \mathbb{N}$ if there is some $b \in \mathbb{N}$ such that $b \geq 2$ and $n = a^b$. For example, 81 is a perfect power of 3, since $81 = 3^4$.

The **logarithm** to base 2 of a real number $x > 0$ is denoted by $\log x$, i.e. $\log x$ is the (unique) real number ℓ that satisfies $2^\ell = x$. For example, $\log 2 = 1$ and $\log 8 = 3$. Once in a while we also use the **natural logarithm** $\ln x$: this is the logarithm to base e, where e is Euler's constant. That is, $\ln x$ is the number $\ell \in \mathbb{R}$ for which $e^\ell = x$.

If N and M are sets, then "$f : N \to M$" is an abbreviation for "f is a function from N to M". This means that f associates to every element $x \in N$ an element of M, usually denoted by $f(x)$. For example, we can define a function f from $\mathbb{N}$ to $\mathbb{R}$ by setting $f(n) := \log n$ for all $n \in \mathbb{N}$.

If $x \in \mathbb{R}$, then $|x|$ denotes the **absolute value** of x. (So $|x| = x$ when $x \geq 0$ and $|x| = -x$ otherwise.) We also write $\lfloor x \rfloor$ to denote the largest integer n with the property that $n \leq x$. Similarly, we write $\lceil x \rceil$ for the smallest integer n satisfying $n \geq x$. (See Exercise 1.1.11.) For example, $\left\lfloor \frac{3}{2} \right\rfloor = 1$ and $\left\lceil \frac{3}{2} \right\rceil = 2$.

Mathematicians often use Greek letters to refer to certain quantities (for example, angles in elementary geometry). We will utilize the uppercase letter Π ("Pi") and the lowercase letters α ("alpha"), δ ("delta"), ε ("epsilon"), ζ ("zeta"), μ ("mu"), φ ("phi"), and, of course, π ("pi").

All other concepts and notations are introduced at the appropriate time (with many examples). They are also listed in the index and in the list of symbols at the end of the book.

Part 1

Foundations

Chapter 1

Natural numbers and primes

Throughout this book, we study natural numbers and the problem of distinguishing between those that are prime and those that are composite. Hence we begin by recalling and proving some of the fundamental properties of these numbers.

We will first learn about the principle of **mathematical induction** and define divisibility. We prove the key result that every natural number can be written uniquely as a product of prime numbers (the “Fundamental Theorem of Arithmetic”) and derive and use the **Euclidean algorithm**, which allows us to tell whether two numbers have a common factor *without* having to compute their prime factor decomposition! To round off the chapter, we study the oldest known primality test – the **Sieve of Eratosthenes** – and also show that there are infinitely many different prime numbers.

1.1. The natural numbers

Let us think of the natural numbers quite naively as those quantities used to count things – whenever we count (finitely many) objects, their number should be an element of $\mathbb{N}$. We take the view that counting makes sense only if there actually is something to count:

the number 0 does *not* belong to $\mathbb{N}$. Hence the first and smallest natural number is 1.

From this intuitive point of view, let us discuss some important properties that distinguish $\mathbb{N}$ from other number systems. At first, we notice many things that *cannot* be done with natural numbers. They cannot always be subtracted from one another without leaving the set of natural numbers (as is possible for integers) nor can we divide them without restriction (as with fractions). Taking arbitrary square roots is certainly not possible either, in contrast to the positive real numbers. But $\mathbb{N}$ does have a striking property that has many useful consequences: the **well-ordering principle**.

1.1.1. Well-Ordering Principle.
Every non-empty subset of $\mathbb{N}$ possesses a smallest element.

We see at once that the well-ordering principle is false for $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$. Indeed, if a is an arbitrary integer, then $a-1$ is also an integer, and $a-1$ is smaller than a. So $\mathbb{Z}$ does not contain a smallest element. (However, the well-ordering principle does hold for subsets of $\mathbb{Z}$ that are *bounded from below*; see Exercise 1.1.11. We will frequently use this fact.)

On the other hand, we can justify the principle for the natural numbers as follows. If A is a non-empty subset of $\mathbb{N}$, then A contains some element $n_0 \in A$. If n_0 is not the smallest element of A, then there is some other element $n_1 \in A$ such that $n_1 < n_0$. If n_1 again is not the smallest element of A, then there is an even smaller one, and so on. But there are only $n_0 - 1$ natural numbers that are smaller than n_0, so the procedure must necessarily come to an end. Thus at some point, we will have found the smallest element of A.

Strictly speaking, this is not a proof, but merely an argument that the well-ordering principle is plausible. Indeed, we cannot give a formal proof because we did not *define* the natural numbers with the necessary accuracy. Instead, we accept the well-ordering principle as an **axiom**, i.e. a proposition whose truth we take to be evident without proof. (However, see Exercise 1.1.24.)

The method of infinite descent. The well-ordering principle provides us with a useful tool for proving statements about all natural numbers. The idea is best illustrated by an example. (Regarding the name of the following theorem, see Comment 1.1.21.)

1.1.2. Theorem (Irrationality of $\sqrt{2}$)**.**
Let n be a natural number. Then there is no natural number m with $2m^2 = n^2$.

Proof. We assume, by way of contradiction, that the claim is wrong; so suppose that there are natural numbers n and m with $2m^2 = n^2$. To see what this would entail, it might make things simpler to choose n and m as small as possible, using the well-ordering principle. Indeed, our assumption means that the set

$$A := \{n \in \mathbb{N} : \text{ there is some } m \in \mathbb{N} \text{ such that } 2m^2 = n^2\}$$

is not empty, so it has a smallest element n_0. Let us try to find out some more things about this number n_0. By definition, there exists $m_0 \in \mathbb{N}$ with

$$2m_0^2 = n_0^2. \tag{1.1.3}$$

Since $1 < 2$, we have $m_0^2 < n_0^2$, and thus m_0 is smaller than n_0.

We also see from (1.1.3) that n_0^2 is an even number. So n_0 must itself be even, since the square of an odd number is odd (Exercise 1.1.10). In other words, there is a number $\tilde{n} \in \mathbb{N}$ with $n_0 = 2\tilde{n}$. Substituting for n_0 in (1.1.3), we see that

$$2m_0^2 = n_0^2 = (2\tilde{n})^2 = 4\tilde{n}^2$$

and therefore $m_0^2 = 2\tilde{n}^2$. We conclude that m_0 also belongs to the set A. But this is impossible because $m_0 < n_0$ and n_0 was chosen to be the smallest element of A! So we have derived a contradiction – it follows that our original premise was false and that the theorem is true. ■

The principle underlying the preceding proof is called **the method of infinite descent** or **the method of the smallest counterexample**. The idea is to assume that there is a natural number for which the claim (i.e. a statement that we wish to prove for all $n \in \mathbb{N}$) is false. By the well-ordering principle, there exists a "smallest counterexample": a smallest natural number that violates our statement. If, by studying the properties of this number, we can deduce that there would have to be an even smaller counterexample, then we obtain a contradiction. Throughout the book, the reader will encounter many applications of this valuable proof principle for the natural numbers.

Mathematical induction. The method of infinite descent is closely related to another method of proof, called **(mathematical) induction**. The following theorem formulates the concept as a general principle; Example 1.1.5 below illustrates how one applies the procedure to a specific problem.

1.1.4. Theorem (Induction principle)**.**
Suppose that $M \subseteq \mathbb{N}$ is a set of natural numbers with the following properties:

(a) *the number 1 is an element of M and*

(b) *if n is a natural number in M, then the next number $n+1$ is also an element of M.*

Then it follows that $M = \mathbb{N}$, i.e. every natural number belongs to M.

Proof. We assume that $M \neq \mathbb{N}$ and deduce a contradiction, using the method of the smallest counterexample. By our assumption, the set $A := \{n \in \mathbb{N} : n \notin M\}$, i.e. the collection of natural numbers that do not belong to M, is a non-empty subset of $\mathbb{N}$. By the well-ordering principle, this set has a smallest element n_0. Our hypothesis (a) implies that 1 is not in A, and in particular $n_0 \neq 1$. Therefore $m := n_0 - 1$ is also a natural number. Since n_0 is the smallest element of A, the number m cannot belong to A and it follows that $m \in M$. But hypothesis (b) shows that $n_0 = m + 1$ must also be an element of M, and this is a contradiction. ■

We can visualize the principle of mathematical induction by imagining an (infinite) row of dominoes, placed in such a way that each domino will, in falling, cause the next one to topple as well. The principle of mathematical induction tells us that, if we push over the first one, every domino will eventually fall. This is certainly supported by intuition!

In order to prove some property for all natural numbers using mathematical induction, we show:

- the number 1 has the desired property (**basis of the induction**) and
- if n has the desired property, then $n+1$ also does (**induction step**).

Then Theorem 1.1.4 tells us that the desired statement does indeed hold for *all* natural numbers.

1.1.5. Example. We demonstrate that, for all natural numbers n, the number $n^3 - n$ is a multiple of 3. (That is, there is an integer m such that $n^3 - n = 3m$.)

Proof. If $n = 1$, then

$$n^3 - n = 1^3 - 1 = 1 - 1 = 0 = 3 \cdot 0,$$

so the claim is true in this case. That is the basis of the induction.

Now let us take a look at an arbitrary natural number n for which the claim is true. Then there is an integer m such that $n^3 - n = 3m$. This is called the **induction hypothesis**.

We need to show that the claim is also true for $n + 1$. That is, we must check that $(n+1)^3 - (n+1)$ is a multiple of 3. To do so, we expand the power $(n + 1)^3$ (see also Theorem 1.1.9) and see that

$$(n + 1)^3 - (n + 1) = n^3 + 3n^2 + 3n + 1 - n - 1 = n^3 + 3n^2 + 3n - n.$$

Note that $n^3 - n$ appears on the right-hand side, and our induction hypothesis tells us that $n^3 - n = 3m$. So

$$(n + 1)^3 - (n + 1) = 3m + 3n^2 + 3n = 3(m + n^2 + n).$$

Since $m + n^2 + n$ is certainly an integer, we have shown that $(n + 1)^3 - (n + 1)$ is a multiple of 3, as desired.

This completes the induction step, and the claim is proved for all natural numbers n. ■

There are a few variants of the induction principle that will be used on occasion:

(a) Sometimes it is convenient to pass from $n-1$ to n in the induction step, rather than from n to $n+1$.

(b) Statements that are true for all integers greater than or equal to some number n_0 can also be proved using mathematical induction. In this case, in the basis of the induction, we simply check the statement for $n = n_0$ (instead of $n = 1$); everything else is exactly as above. In particular, if we would like to have a formula available for all numbers in $\mathbb{N}_0$, rather than all numbers in $\mathbb{N}$, we start the induction at $n = 0$.

(c) In some cases, we derive the desired property for $n+1$ using not only the induction hypothesis for n, but also for $n-1$, or even for all $m \leq n$. That is, the principle of mathematical induction still applies when the induction hypothesis takes the stronger form "We suppose that the claim is true for all smaller numbers"; see Exercise 1.1.25.

Since we deduced the principle of mathematical induction from the well-ordering principle, every inductive proof can also be formulated as a proof by infinite descent (and vice versa; see Exercise 1.1.24). However, induction yields a **direct** proof (i.e. one that does not rely on finding contradictions), which is often more elegant. We shall decide on a case-by-case basis which method to use, depending on the specific application and also on personal taste.

Recursive definitions. Using mathematical induction, we can define certain sequences $a_1, a_2, a_3, \ldots$ of numbers without having to explicitly write down a formula. Indeed, suppose that we specify the value of the first number a_1 and know how to obtain a_{k+1} from the numbers $a_1, \ldots, a_k$. Then there is enough information to determine a_k uniquely for all $k \in \mathbb{N}$. This is called a **recursive definition**.

For example, we define a sequence $a_1, a_2, ...$ via

$$a_1 := 1 \quad \text{and} \quad a_{k+1} := \frac{1}{1+a_k} \text{ for all } k \geq 1. \tag{1.1.6}$$

From this information we can compute all numbers in the sequence, starting with a_1:

$$\begin{aligned} &a_1 = 1, && a_2 = \frac{1}{1+a_1} = \frac{1}{2}, && a_3 = \frac{1}{1+a_2} = \frac{2}{3}, \\ &a_4 = \frac{1}{1+a_3} = \frac{3}{5}, && a_5 = \frac{1}{1+a_4} = \frac{5}{8}, && a_6 = \frac{1}{1+a_5} = \frac{8}{13}, \end{aligned}$$

and so on. It is often possible to use mathematical induction to prove statements about a recursively defined sequence of number, even if we do not know an explicit formula for the k-th element.

The **Fibonacci numbers** are an important example of a recursively defined sequence. They are given by

$$f_1 := 1; \quad f_2 := 1; \quad f_k := f_{k-1} + f_{k-2} \text{ for all } k \geq 3. \tag{1.1.7}$$

The first few terms are 1, 1, 2, 3, 5, 8, 13, ...

Finally, we mention the **binomial coefficients**, from the mathematical field of **combinatorics**, which play an important role in this book on a number of occasions. If $n, k \in \mathbb{N}$, then the binomial coefficient $\binom{n}{k}$ is defined to be the number of ways of choosing k out of n different-colored balls, without repetitions and disregarding the order in which the balls are picked. (In other words, $\binom{n}{k}$ is *the number of subsets of* $\{1, 2, \ldots, n\}$ that contain exactly k elements.) For example, in the United Kingdom's National Lottery there are exactly $\binom{49}{6}$ different possibilities of picking 6 different numbers from the range 1–49. Binomial coefficients satisfy the following recursive formula:

$$\binom{n}{0} = 1, \quad \binom{0}{k} = 0, \quad \text{and} \quad \binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k} \tag{1.1.8}$$

for all $n \in \mathbb{N}_0$ and all $k \in \mathbb{N}$. (See Exercise 1.1.15.) In the following, we use this as a formal (recursive) definition of binomial coefficients. **Pascal's triangle** (Figure 1.1) is a succinct way of illustrating the formula (1.1.8) visually.

Binomial coefficients appear, in particular, when multiplying out powers of a sum of two elements. This is the content of the **binomial theorem**.

$$\begin{array}{ccccccccccccc}
 & & & & & & 1 & & & & & & \\
 & & & & & 1 & & 1 & & & & & \\
 & & & & 1 & & 2 & & 1 & & & & \\
 & & & 1 & & 3 & & 3 & & 1 & & & \\
 & & 1 & & 4 & & 6 & & 4 & & 1 & & \\
 & 1 & & 5 & & 10 & & 10 & & 5 & & 1 & \\
1 & & 6 & & 15 & & 20 & & 15 & & 6 & & 1 \\
\multicolumn{13}{c}{\cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots}
\end{array}$$

Figure 1.1. The first rows of Pascal's triangle. The $(n+1)$-th row contains the binomial coefficients $\binom{n}{0}, \ldots, \binom{n}{n}$. By (1.1.8), each entry is obtained by adding the two entries diagonally above it.

1.1.9. Theorem (Binomial theorem)**.**
Let a and b be arbitrary real numbers and let $n \geq 0$ be an integer. Then

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}.$$

Remark. For $n = 2$, we obtain $(a+b)^2 = a^2+2ab+b^2$, the well-known formula for the square of a sum. In Example 1.1.5, multiplying out $(n+1)^3 = n^3 + 3n^2 + 3n + 1$ corresponds to the case $n = 3$.

Proof of the binomial theorem. It is a useful exercise to justify the claim using the combinatorial definition of binomial coefficients. (How many times does the term $a^k b^{n-k}$ appear when multiplying out $(a+b)^n$?) Instead, we use the opportunity to practice mathematical induction with a slightly more complicated example than before.

Let us keep the numbers a and b fixed. First suppose that $n = 0$; then $(a+b)^n = 1$ and

$$\sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k} = \binom{0}{0} a^0 b^0 = 1,$$

so the theorem is true in this case. That is the basis of the induction.

Now we suppose that the claim is true for n and we must show that it also holds for $n+1$. Using our induction hypothesis and then multiplying out, we see that

$$\begin{aligned}(a+b)^{n+1} &= (a+b)\cdot(a+b)^n\\ &= (a+b)\sum_{k=0}^{n}\binom{n}{k}a^k b^{n-k}\\ &= \sum_{k=0}^{n}\binom{n}{k}a^{k+1}b^{n-k} + \sum_{k=0}^{n}\binom{n}{k}a^k b^{n+1-k}.\end{aligned}$$

Substituting $j = k+1$ in the first sum and $j = k$ in the second, we obtain

$$(a+b)^{n+1} = \sum_{j=1}^{n+1}\binom{n}{j-1}a^j b^{n+1-j} + \sum_{j=0}^{n}\binom{n}{j}a^j b^{n+1-j}.$$

(We remind the reader that the Σ-notation for sums is merely an abbreviation, so it does not matter what name we give to the summation variable!)

Now separate the last term of the first sum and the first term of the second sum; then we can combine the remaining terms of both. By definition, $\binom{n}{0} = \binom{n}{n} = 1$, so

$$\begin{aligned}(a+b)^{n+1} &= \binom{n}{n}a^{n+1} + \binom{n}{0}b^{n+1}\\ &\quad + \sum_{j=1}^{n}\binom{n}{j-1}a^j b^{n+1-j} + \sum_{j=1}^{n}\binom{n}{j}a^j b^{n+1-j}\\ &= a^{n+1} + b^{n+1} + \sum_{j=1}^{n}\left(\binom{n}{j-1} + \binom{n}{j}\right)a^j b^{n+1-j}.\end{aligned}$$

Now we can use the recursive formula (1.1.8):

$$\begin{aligned}(a+b)^{n+1} &= a^{n+1} + b^{n+1} + \sum_{j=1}^{n}\binom{n+1}{j}a^j b^{n+1-j}\\ &= \sum_{j=0}^{n+1}\binom{n+1}{j}a^j b^{n+1-j}.\end{aligned}$$

This completes the induction step. By mathematical induction, the binomial theorem holds for all n. ■

Exercises.

1.1.10. Exercise (!). Recall that a natural number n is called **even** if there is a natural number m with $n = 2m$. The number n is called **odd** if there is a natural number m such that $n = 2m - 1$.

(a) Show that every natural number is either even or odd, but not both at the same time. (*Hint:* By the well-ordering principle, there is a smallest number m satisfying $2m \geq n$.)

(b) Show that the product of two even numbers is even and that the product of two odd numbers is odd. What can you say about the product of an odd number and an even number?

1.1.11. Exercise (!). Suppose that M is a non-empty set of integers. Then M is called **bounded from above** or **bounded from below** if there exists an integer K such that $x \leq K$ for all $x \in M$ or $x \geq K$ for all $x \in M$, respectively.

Prove the following: if M is bounded from below, then M contains a smallest element. Similarly, if M is bounded from above, then M has a largest element. Are these statements also true for subsets of $\mathbb{Q}$ or $\mathbb{R}$?

(*Hint:* For the first part apply the well-ordering principle to the set of all numbers of the form $1 + x - K$, with $x \in M$. For the second part, apply the first claim to the set $\{x \in \mathbb{Z} : -x \in M\}$.)

1.1.12. Exercise (!). Prove by mathematical induction for all $n \in \mathbb{N}$:

(a) $2^n \geq 2n$.

(b) $\displaystyle\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$ ("Gauss summation").

(c) $\displaystyle\sum_{k=0}^{n-1} x^k = \frac{1 - x^n}{1 - x}$ for all real numbers $x \neq 1$.

(d) For all real numbers $x \neq 1$,
$$\sum_{k=0}^{n-1} (k+1) \cdot x^k = \frac{nx^{n+1} - (n+1)x^n + 1}{(1-x)^2}.$$

(e) $\displaystyle\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$.

(f) $\displaystyle\sum_{k=0}^{n} (k \cdot k!) = (n+1)! - 1$.

(g) $\displaystyle\sum_{k=1}^{n} \frac{1}{k(k+1)} = \frac{n}{n+1}$.

1.1.13. Exercise.

(a) Prove that $n^5 - n$ is a multiple of 5 for all $n \in \mathbb{N}$.

(b) Is $n^4 - n$ a multiple of 4 for all $n \in \mathbb{N}$? If not, give a counterexample and explain why the proof from (a) fails here.

1.1.14. Exercise. Suppose that $g_1, \dots, g_n$ are straight lines in the plane, no two of which are parallel to each other. (We say that the lines are **pairwise** not parallel.) Furthermore no more than two lines should intersect at any given point in the plane.

Into how many different pieces do these lines separate the plane? Develop an idea, write down a formula, and prove it using mathematical induction. What changes if some of the lines are allowed to be parallel?

1.1.15. Exercise (!).

(a) Using the intuitive definition of binomial coefficients, prove that the recursive formula (1.1.8) is true.

(b) Also explain why the equation

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{1.1.16}$$

holds for all $k, n \in \mathbb{N}_0$ with $k \leq n$.

(c) Deduce (1.1.16) from the recursive formula (1.1.8), using mathematical induction.

1.1.17. Exercise (!). Let $n, k, \ell \in \mathbb{N}_0$. Prove the following:

(a) $\binom{n+\ell}{k} \geq \binom{n}{k}$.

(b) $\binom{n+\ell}{k+\ell} \geq \binom{n}{k}$.

(c) The “middle” binomial coefficients $\binom{2n}{n}$ grow at least exponentially, i.e.

$$\binom{2n}{n} \geq 2^n.$$

(*Hint:* For the first two parts it is useful to take a look at Pascal’s triangle. For the third part, use induction together with the recursive formula for binomial coefficients and the first two parts of the exercise.)

1.1.18. Exercise (!). Let $n, k \in \mathbb{N}_0$ and write $a(n, k)$ for the number of ways of choosing up to k (not necessarily different) numbers from 1 to n, disregarding the order in which they are picked. (Here we allow the possibility of not picking *any* numbers; for example $a(n,0) = 1$ and $a(n,1) = n+1$ for all n.)

(a) Justify that, for all natural numbers $n, m \geq 1$, the number $a(n,m)$ satisfies the recursive formula

$$a(n,m) = a(n-1,m) + a(n,m-1).$$

(b) Prove by induction that

$$a(n,m) = \binom{n+m}{m}.$$

1.1.19. Exercise. Prove that the rational numbers a_k defined by (1.1.6) satisfy $a_k = f_k/f_{k+1}$ for all k, where f_k is the k-th Fibonacci number as defined in (1.1.7).

1.1.20. Exercise. Show that the k-th Fibonacci number is given by

$$f_k = \frac{(1+\sqrt{5})^k - (1-\sqrt{5})^k}{2^k \cdot \sqrt{5}} \qquad \text{(for all } k \geq 1\text{)}.$$

(*Hint:* Use version (c) of the induction principle.)

Further Exercises and Comments.

1.1.21. Theorem 1.1.2 is equivalent to the well-known fact that $\sqrt{2}$ is an irrational number. Indeed, $2m^2 = n^2$ just means that

$$\left(\frac{n}{m}\right)^2 = 2.$$

There also is a geometric interpretation of this statement: there is no square for which both the sidelength a and the length d of the diagonal are natural numbers. (Otherwise, Pythagoras's Theorem implies that $2a^2 = d^2$.)

1.1.22. If n is a natural number, then $n+1$ is usually called the **successor** of n. The following are evident:

(I) 1 is a natural number;

(II) every natural number has exactly one successor;

(III) there is no natural number whose successor is 1, but every natural number $n \neq 1$ is itself the successor of some other natural number;

(IV) different natural numbers have different successors;

(V) if M is a subset of $\mathbb{N}$ that contains 1 and also contains the successor of each of its elements, then $M = \mathbb{N}$.

Here statement (V) is exactly the induction principle proved in Theorem 1.1.4. Properties (I) to (V) are known as the **Peano axioms**, after the Italian mathematician Giuseppe Peano.

It turns out that the Peano axioms describe the natural numbers uniquely, i.e. there is (up to a relabeling of the elements) no other set with the same properties. For this reason, they can be used to *define* the natural numbers; this is the usual way of introducing them in a university-level mathematics course. We decided to begin with the well-ordering principle instead, feeling that this would appear more intuitive to readers who have not encountered either concept before.

1.1.23. Exercise. Note that the Peano axioms do not include any statements about elementary arithmetic; they require only that for every natural number n the successor $n + 1$ is defined.

Show that, using this successor function, the sums $n+m$, products $n\cdot m$, and powers n^m for all natural numbers n, m can be defined recursively.

1.1.24. Exercise. Show that the well-ordering principle follows from the principle of mathematical induction (and hence from the Peano axioms).

1.1.25. Exercise. Prove – using either the well-ordering principle or the Peano axioms – that an analogue of Theorem 1.1.4 holds for each of the alternative versions of the induction principle introduced in the text.

1.1.26. The proof principle of infinite descent was first described by Pierre de Fermat (1601–1665), a French lawyer who intensively studied mathematics in his spare time. He did not publish mathematical texts himself; we know about his work from the letters he exchanged with other mathematicians and from handwritten notes that he made in the margins of textbooks. Fermat is famous for his claim that, for all natural numbers $n > 2$, there are no natural numbers a, b, and c such that $a^n + b^n = c^n$.

Although it is believed today that Fermat did not know of a correct proof of this statement, it has become known as **Fermat's Last Theorem**. It took 300 years until the problem was finally solved by the English mathematician Andrew Wiles. (His proof appeared in the *Annals of Mathematics* in 1995.)

1.1.27. Both Example 1.1.5 and Exercise 1.1.13(a) are special cases of **Fermat's Little Theorem**, which we will encounter in Section 3.2.

1.1.28. Exercise. We claim that every natural number can be described by a sentence containing at most two hundred letters.

"Proof": Assume that the claim is false. Then there exists *the smallest natural number that cannot be described by a sentence containing at most two hundred letters.* This description contains less than two hundred letters – a contradiction!

However, the claim above is clearly wrong. Indeed, there are only finitely many sentences containing at most two hundred letters, but infinitely many natural numbers .... Where did we make a mistake?

1.2. Divisibility and primes

Having discussed the natural numbers, we are now ready to discuss the question of when we can divide one natural number by another, which naturally leads to the definition of prime numbers. We begin by formally introducing a few concepts with which the reader is likely to be familiar.

1.2.1. Definition (Divisors, multiples, and primes)**.**
Let $n, k \in \mathbb{Z}$. We call k a **divisor** of n (and conversely n a **multiple** of k) if there exists an integer m such that $k \cdot m = n$. In this case, we say that k **divides** n, or that n is **divisible** by k, and write $k \,|\, n$.

A **prime** is a natural number that is divisible by exactly two different natural numbers (namely by 1 and itself).

A natural number $n > 1$ that is not prime is called a **composite number**.

For example, $3 \,|\, 6$ because $3 \cdot 2 = 6$. We also have $-3 \,|\, 6$, since $(-3)\cdot(-2) = 6$. Every integer is divisible by both 1 and -1. Similarly, every integer is a divisor of zero. (This is a good reason to leave zero out of the natural numbers.) If a, b are natural numbers and a divides b, then we see immediately that $a \leq b$. Readers are invited to convince themselves that this is not true when a and b are allowed to be negative!

If $n \in \mathbb{Z}$, then 1, -1, n, and $-n$ are certainly divisors of n; they are called the **trivial divisors**. If there are any other divisors, then they are called **non-trivial divisors**. Thus prime numbers only have trivial divisors. Note that 1 is *not* prime, since there is only one natural number that divides it! The next result provides us with some simple rules for working with divisors.

1.2.2. Theorem (Rules for divisibility)**.**
Let $a, b, c \in \mathbb{Z}$. If b and c are divisible by a, then so are $b + c$, $b - c$, and $b \cdot c$. Every divisor of b divides every multiple of b; that is, if $a \mid b$ and $b \mid c$, then also $a \mid c$.

Proof. Suppose that a divides both b and c. Then, by definition, there exist integers n and m such that $a \cdot n = b$ and $a \cdot m = c$. Hence

$$\begin{aligned} b + c &= a \cdot n + a \cdot m = a \cdot (n + m), \\ b - c &= a \cdot n - a \cdot m = a \cdot (n - m), \\ b \cdot c &= (an) \cdot (am) = a \cdot (n \cdot a \cdot m). \end{aligned}$$

Since $n + m$, $n - m$, and $n \cdot a \cdot m$ are integers, it follows that a is a divisor of $b + c$, of $b - c$, and of $b \cdot c$. The final claim follows similarly; we leave the details to the reader as an exercise. ■

1.2.3. Definition (Common divisors and multiples; gcd and lcm)**.**
Let $a, b \in \mathbb{Z}$ and let $k \in \mathbb{Z}$ be a number that divides both a and b. Then k is called a **common divisor** or a **common factor** of a and b. Similarly, a number $v \in \mathbb{Z}$ that is a multiple of both a and b is called a **common multiple** of a and b.

Now suppose that $a \neq 0$ or $b \neq 0$. Then the largest number $k \in \mathbb{N}$ that is a common divisor of a and b is referred to as the **greatest common divisor** (gcd) of a and b. We write $k = \gcd(a, b)$. (The gcd is sometimes also called the **"highest common factor"**.)

Correspondingly, if neither a nor b is equal to zero, the smallest natural number v that is a common multiple of a and b is called the **least common multiple** (lcm) of a and b; we write $v = \operatorname{lcm}(a, b)$. For completeness, we also define $\gcd(0, 0) := 0$ and $\operatorname{lcm}(a, 0) := \operatorname{lcm}(0, a) := 0$ for all $a \in \mathbb{Z}$.

Two integers a and b are called **coprime** if $\gcd(a, b) = 1$, i.e. if a and b do not have any positive common divisors apart from 1.

Remarks.

(a) The least common multiple exists by the well-ordering principle. Furthermore, for every $c \neq 0$ the set of divisors of c is bounded from above by the absolute value $|c|$. Hence the gcd also always exists (see Exercise 1.1.11).

(b) From school, we know that every common divisor of a and b also divides $\gcd(a,b)$. This is not completely obvious from the definition and therefore requires a proof, which we shall give in the next section. A simple method of computing $\gcd(a,b)$ will be introduced in Section 1.4.

Examples.

(a) If $n \in \mathbb{Z}$ is arbitrary, then $\gcd(1,n) = 1$ and $\gcd(0,n) = n$.

(b) The positive common divisors of 12 and 18 are 1, 2, 3, and 6. Hence $\gcd(12,18) = 6$.

(c) The numbers 3 and -6 are *not* coprime as $\gcd(3,-6) = 3$. In contrast, $\gcd(5,12) = \gcd(5,17) = \gcd(12,17) = 1$, so the numbers 5, 12, and 17 are pairwise coprime.

We now turn to the concept of **division with remainder**. Since this is central for our book and for number theory in general, we shall prove formally that division with remainder is always possible. This is sometimes referred to as the **“division algorithm”**; however, it is a theorem, not an algorithm in the sense of our book.

1.2.4. Theorem (Division theorem)**.**
Let $a \in \mathbb{Z}$ and $b \in \mathbb{N}$. Then there exist integers q and r such that $0 \leq r < b$ and $a = qb + r$.

*We say that b **divides** a **with remainder** r. The numbers q and r are uniquely determined by a and b.*

Remark. In particular, b divides a with remainder 0 if and only if b is a divisor of a.

Proof. The idea is simple: we choose q as large as possible while requiring that the remainder $r = a - qb$ is not negative. Then q

and r have the desired property and are uniquely determined by this description.

To develop this outline into a formal proof, consider the set $Q := \{q \in \mathbb{Z} : a - qb \geq 0\}$. We have

$$a - (-|a| \cdot b) = a + |a| \cdot b \geq a + |a| \geq 0,$$

so $-|a| \in Q$ and in particular $Q \neq \varnothing$. On the other hand, we have $a - nb < 0$ for all $n > |a|$, so it follows that the set Q is bounded from above. By Exercise 1.1.11, Q has a largest element q. We set $r := a - qb \geq 0$. By maximality of q, we see that $q + 1 \notin Q$ whence

$$r - b = a - qb - b = a - (q + 1)b < 0.$$

Hence it follows that $0 \leq r < b$ and $a = qb + r$ as claimed.

To prove uniqueness, let us assume that q', r' are integers such that $a = q'b + r'$ with $0 \leq r' < b$. We need to show that $q' = q$ and $r' = r$. By definition of the set Q, we know that $q' \in Q$. Also, we see for all $n \geq q' + 1$ that

$$a - nb \leq a - (q' + 1)b = a - q'b - b = r' - b < 0.$$

Therefore $n \notin Q$. This means that q' is the largest element of Q, giving $q' = q$. But then we also have $r' = a - q'b = a - qb = r$, as desired. ■

Example. The number 5 divides 47 with remainder 2, as $47 = 9 \cdot 5 + 2$ and $0 \leq 2 < 5$. For larger examples, the numbers q and r can be found with the usual method of **long division**. For example, to divide 10 007 by 101:

$$\begin{array}{r} 99 \\ 101\,\overline{)\,10007} \\ \underline{9090} \\ 917 \\ \underline{909} \\ 8 \end{array}$$

So $10\,007 = 99 \cdot 101 + 8$, and 101 divides 10 007 with remainder 8.

In Section 3.1, we will discuss many further properties of division with remainder. However, the concept will already be important for developing the **Euclidean algorithm** in Section 1.4.

Exercises.

1.2.5. Exercise. Suppose that n and m are integers and let $k \in \mathbb{N}$. Prove or disprove the following statements!

(a) If k divides $m + n$, then it also divides n and m.

(b) If k divides $m \cdot n$, then k is also a divisor of n and m.

(c) If k is a divisor of $m \cdot n$, then k divides n or m.

(d) If k divides m but not n, then k does not divide $m + n$.

(e) If k divides m, but not n, then k does not divide $m \cdot n$.

(f) If k divides n and m with remainder 1, then the remainder of $n \cdot m$ after division by k is also 1.

(g) If k divides both n and m with remainder 1, then k also divides $n + m$ with remainder 1.

1.2.6. Exercise (!).

(a) Show that every natural number $n > 1$ has at least one **prime factor**. (This means that there exists a prime number p such that $p \,|\, n$.)

(b) Show that a composite number $n > 1$ has at least one non-trivial divisor k such that $k^2 \leq n$.

1.2.7. Exercise. Let $k \in \mathbb{N}$ and $a, b \in \mathbb{Z}$. Furthermore, suppose that k divides the number a with remainder r and the number b with remainder s. Develop rules for the remainders of $a + b$, $a - b$, and $a \cdot b$ when divided by k, and then prove the correctness of these rules!

1.2.8. Exercise. Let $n \in \mathbb{N}$. Show that exactly one of the numbers n, $n + 1$, and $n + 2$ is divisible by 3.

1.2.9. Exercise. Let $n \geq 3$ be an odd natural number. Show that exactly one of the numbers $n + 1$ and $n - 1$ is divisible by 4.

1.2.10. Exercise. Let a, b, c, and d be integers such that $a \,|\, b$ and $c \,|\, d$. Does this imply that $ac \,|\, bd$?

1.2.11. Exercise. Show that, for all integers a and b, the number $2a + b$ is divisible by 7 if and only if $100a + b$ is divisible by 7. Use this fact to decide whether 100002 is divisible by 7. Can you find and prove similar “division rules”?

1.2.12. Exercise (!). Let $n \in \mathbb{N}$ and let p be prime. Prove: if p does not divide n, then n and p are coprime.

1.3. Prime factor decomposition

The importance of prime numbers stems from the fact that they are in some sense the "building blocks" of natural numbers. Indeed, every natural number $n \geq 2$ has a **prime factor decomposition**[1]; i.e. it can be written as a product of prime numbers

$$n = p_1 \cdot p_2 \cdots p_k. \tag{1.3.1}$$

(These numbers $p_1, \dots, p_k$ are called the **prime factors** of n.) Furthermore, the decomposition is unique – apart from the possibility of reordering the factors in (1.3.1). We now formally prove these facts, together called the **Fundamental Theorem of Arithmetic**.

1.3.2. Theorem (Fundamental Theorem of Arithmetic)**.**
Let $n \geq 2$ be a natural number. Then n is a product of prime numbers, and this decomposition into primes is unique up to a reordering of the factors.

(In particular, the number of distinct primes and their multiplicities in the prime factor decomposition are uniquely determined.)

Proof. We shall prove the theorem using variant (c) of the induction principle. That is, let $n_0 \geq 2$ be a natural number, and suppose that we know that all smaller numbers $n \in \{2, 3, \dots, n_0 - 1\}$ have a unique prime factor decomposition. For convenient notation, let us denote this decomposition by $\Pi(n)$; we must show that n_0 also has a unique prime factor decomposition.

First observe that the claim is true when n_0 is prime: in this case the decomposition consists of a single prime number and is unique because no other prime divides n_0.

Now suppose that n_0 is not prime, and let p be the smallest prime divisor of n_0 (see Exercise 1.2.6). Then we can write $n_0 = p \cdot k$ for some k with $2 \leq k < n_0$. By the induction hypothesis, k has a unique

[1] Sometimes it is useful to agree by convention that $n = 1$ also has a prime factor decomposition, namely the *empty* decomposition into no prime factors. This makes some statements and proofs easier because it is not necessary to treat this case differently.

prime factor decomposition $\Pi(k)$. In particular, n_0 has the prime factor decomposition

$$n_0 = p \cdot \Pi(k),$$

and this is the unique decomposition (up to reordering) that contains the number p. Consider any decomposition

$$n_0 = p_1 \cdot p_2 \cdots p_m,$$

where the prime factors are written in non-decreasing order. Recall that p is the *smallest* prime divisor of n_0, so $p_1 \geq p$; we must show that $p_1 = p$.

Let us assume, by contradiction, that $p_1 > p$. Then we divide p_1 by p with remainder; hence we let $q, r \in \mathbb{Z}$ be such that

$$p_1 = q \cdot p + r$$

and $0 \leq r < p$. Since p and p_1 are distinct prime numbers by assumption, we must have $r \geq 1$. Now we can write

$$n_0 = (q \cdot p + r) \cdot p_2 \cdots p_m = p \cdot q \cdot \ell + r \cdot \ell, \tag{1.3.3}$$

where we abbreviate $\ell = p_2 \cdots p_m$. The number $r \cdot \ell$ has the prime factor decomposition

$$r \cdot \ell = \Pi(r) \cdot p_2 \cdots p_m. \tag{1.3.4}$$

But p divides both n_0 and $p \cdot q \cdot \ell$, so according to (1.3.3), $r \cdot \ell$ is divisible by p. By the induction hypothesis, this means that p must appear in the prime factor decomposition of $r \cdot \ell$. But p is not visible in (1.3.4)! (Recall that $r < p$.) This is a contradiction. The induction, and hence the proof of the theorem, is complete. ■

Looking a little more closely at the proof, we notice that it is *uniqueness* of the decomposition, rather than existence, that posed the greatest difficulty. More precisely, we needed to show that a prime divides the product of several integers only if it divides one of these integers themselves. As this observation is important in its own right, we shall record it here for further reference.

1.3.5. Corollary (Prime divisors of a product)**.**
Let a and b be integers and let p be a prime divisor of the product $a \cdot b$. Then p divides a or b.

More generally, suppose that $a, b \in \mathbb{Z}$ are both coprime to $k \in \mathbb{Z}$. Then the product $a \cdot b$ is also coprime to k.

Proof. To prove the first claim, we can assume that a and b are both natural numbers ≥ 2 (since the divisors of a are precisely the divisors of $|a|$, and the claim is trivial when one of a and b is zero or one). By hypothesis let $k \in \mathbb{N}$ be such that $p \cdot k = a \cdot b$.

Let us write again $\Pi(n)$ for the prime factor decomposition of a natural number $n \geq 2$; then $a \cdot b$ has the decompositions

$$a \cdot b = p \cdot \Pi(k) \qquad \text{and} \qquad a \cdot b = \Pi(a) \cdot \Pi(b).$$

By the Fundamental Theorem of Arithmetic, the two decompositions agree up to reordering. So p must occur in $\Pi(a)$ or in $\Pi(b)$, as claimed.

The second claim follows from the first. Indeed, suppose that $a \cdot b$ is *not* coprime to k, and let p be a prime divisor of $\gcd(a \cdot b, k)$. Then p divides both k and $a \cdot b$. By the first part of the corollary, one of a and b is a multiple of p, and hence not coprime to k. ■

More generally, a natural number k divides another number n if and only if the prime factor decomposition of k is contained in the prime factor decomposition of n. So, if n and m are natural numbers, we can find the greatest common divisor and the lowest common multiple of n and m from their decompositions. For example, $n = 90$ and $m = 315$ can be written as $n = 2 \cdot 3 \cdot 3 \cdot 5$ and $m = 3 \cdot 3 \cdot 5 \cdot 7$, so

$$\gcd(n, m) = 3 \cdot 3 \cdot 5 = 45 \quad \text{and} \quad \operatorname{lcm}(n, m) = 2 \cdot 3 \cdot 3 \cdot 5 \cdot 7 = 630.$$

This representation of gcd and lcm using prime factors will be extremely helpful for us in several places. For example, it immediately implies an important property of the gcd:

1.3.6. Theorem (Divisors of the gcd)**.**
Let $a, b \in \mathbb{Z}$. Then every common divisor of a and b also divides $\gcd(a, b)$.

On the other hand, it is very difficult to find the prime factor decomposition of a large integer, so computing the gcd and lcm in this manner is not practical! In the next section, we develop a much more effective method. (Compare Exercises 1.3.8 and 1.4.5.)

Exercises.

1.3.7. Exercise. Find the prime factor decomposition of the numbers 600, 851, and 1449.

1.3.8. Exercise. Compute $\gcd(1961, 1591)$ by finding the prime factor decomposition of the two numbers.

1.3.9. Exercise (!). Let $a, b \in \mathbb{Z}$ and $d := \gcd(a, b)$. Prove:

(a) $\frac{a}{d}$ and $\frac{b}{d}$ are coprime.

(b) If v is a common multiple of a and b, then v is a multiple of $\operatorname{lcm}(a, b)$.

(c) $\operatorname{lcm}(a, b) \cdot d = |a \cdot b|$. In particular, if a and b are coprime, then $\operatorname{lcm}(a, b) = |a \cdot b|$.

1.3.10. Exercise. Let a, b, and c be integers such that c is a divisor of the product $a \cdot b$. Is it true that c divides $\gcd(a, c) \cdot \gcd(b, c)$?

Further Exercises and Comments.

1.3.11. The Fundamental Theorem of Arithmetic can be found implicitly already in Euclid's *Elements*. However, the theorem was not explicitly formulated and proved until 1801, when Gauss did so for the first time in his *Disquisitiones Arithmetica*. For a detailed overview over the history of this theorem we refer the reader to the article [**AÖ**].

1.3.12. Often the gcd is defined to be a positive common divisor that is divisible by every other common divisor (and thus is a "biggest" common factor in that sense). Theorem 1.3.6 is then needed to show that "the" gcd is unique.

1.3.13. Using the results of the next section, we can give an alternative proof of the Fundamental Theorem of Arithmetic that is perhaps more elegant, but less direct, than the one we gave here. (See Exercise 1.4.9.)

1.4. The Euclidean algorithm

So how do we find the greatest common divisor of two given integers a and b? As already mentioned, simply computing all divisors of a and b, or using their prime factor decompositions, is not workable for very large numbers. On the other hand, the **Euclidean algorithm**, which we shall now describe, can be performed very quickly even when a and b have thousands of digits. The development of this method will also give us some important theoretical insights. (Regarding the word "algorithm" and whether or not it is appropriate in this context, we refer the reader to Chapter 2.)

The basic idea is to use a and b to come up with new (and potentially smaller) numbers that have the same common divisors:

1.4.1. Lemma (Pairs of numbers with the same common divisors)**.** *Let a, b, and m be arbitrary integers. Then every common divisor of a and b is also a common divisor of a and $c := b + m \cdot a$; conversely each common divisor of a and c is a common divisor of a and b.*

In particular, $\gcd(a, b) = \gcd(a, b + m \cdot a)$.

Proof. If k is a common divisor of a and b, then, by Theorem 1.2.2, k divides $m \cdot a$, and hence also $c = b + m \cdot a$, as claimed.

On the other hand, let k be a common divisor of a and c. We can apply the fact we just proved to the numbers a, c, and $-m$. This shows that k is a common divisor of a and $c + (-m) \cdot a = (b + m \cdot a) - m \cdot a = b$, as desired. The final statement of the lemma follows from the definition of the greatest common divisor. ■

How do we use this observation? If $a, b \in \mathbb{N}$ are natural numbers with $a > b$, then we can divide a by b with remainder and hence find $q, r \in \mathbb{Z}$ such that $a = q \cdot b + r$ and $0 \leq r < b$. Lemma 1.4.1 tells us that $\gcd(a, b) = \gcd(b, r)$ – so we have reduced the problem to finding the greatest common divisor of another pair of numbers. If

$r \neq 0$, then we can do the same thing again and divide b by r with remainder. Note that, in each step, the remainder becomes strictly smaller. Thus, at some point, we must obtain zero – and can simply read off the gcd!

Let us consider an example: what is the greatest common divisor of 250 and 36? We divide 250 by 36 with remainder:

$$250 = 6 \cdot 36 + 34;$$

so $\gcd(250, 36) = \gcd(36, 34)$. Next we divide 36 by 34:

$$36 = 1 \cdot 34 + 2.$$

We are left with the numbers 34 and 32 and since 34 is divisible by 2, we see that $\gcd(250, 36) = \gcd(36, 34) = \gcd(34, 2) = 2$.

We now formulate this method for general numbers a and b.

```
EUCLIDEAN ALGORITHM
Input: Two numbers $a, b \in \mathbb{Z}$.
  1. If $|a| \geq |b|$, set $r_0 := |a|$ and $r_1 := |b|$; otherwise set $r_0 := |b|$ and $r_1 := |a|$.
  2. Set $j := 1$.
  3. If $r_j = 0$, then $\gcd(a, b) = r_{j-1}$ and we are finished.
  4. Otherwise divide $r_{j-1}$ by $r_j$ with remainder:
         $r_{j-1} = q_j \cdot r_j + r_{j+1}$
     with $q_j, r_{j+1} \in \mathbb{Z}$ and $0 \leq r_{j+1} < r_j$.
  5. Replace $j$ by $j + 1$ and return to Step 3.
```

Since $0 \leq r_{j+1} < r_j$ for all j, there must be some j_* for which we obtain $r_{j_*} = 0$ and finish in Step **3**. So, for every pair of integers a and b, the method will stop eventually. Let $r_0, \ldots, r_{j_*}$ be the numbers obtained while performing the algorithm. By repeated application of Lemma 1.4.1, we see that for all j (ranging from 1 to j_*) the common divisors of r_{j-1} and r_j are exactly the common divisors of a and b. In particular, we have

$$\gcd(a, b) = \gcd(r_{j_*-1}, 0) = r_{j_*-1},$$

which shows that the algorithm really computes the greatest common divisor of a and b (as claimed in Step **3**).

More generally, the common divisors of a and b are precisely the divisors of r_{j_*-1}, so, without noticing, we have found an alternative proof of Theorem 1.3.6!

Another interesting observation is that the Euclidean algorithm not only yields the number $\gcd(a, b)$ but also provides a representation of this number in terms of a and b. For example, take a look at the equations that appeared when we applied the Euclidean algorithm for $a = 250$ and $b = 36$. We see that

$$\begin{aligned}\gcd(a, b) = \gcd(250, 36) = 2 = 36 - 1 \cdot 34 &= 36 - 1 \cdot (250 - 6 \cdot 36)\\ &= 7 \cdot 36 - 250 = (-1) \cdot a + 7 \cdot b.\end{aligned}$$

We can state this as a general principle.

1.4.2. Theorem (Bézout's Lemma)**.**
Let $a, b \in \mathbb{Z}$. Then there are numbers $s, t \in \mathbb{Z}$ such that

$$\gcd(a, b) = s \cdot a + t \cdot b.$$

Proof. Let $r_0, \ldots, r_{j_*}$ be the numbers that appear when the Euclidean algorithm is applied to a and b, as above. We claim: for every $j \in \{0, \ldots, j_*\}$ there are numbers $s_j, t_j \in \mathbb{Z}$ such that $r_j = s_j \cdot a + t_j \cdot b$. As $\gcd(a, b) = r_{j_*-1}$, the proof is complete if we can verify this claim.

Now we proceed by induction. For r_0 and r_1, i.e. for $|a|$ and $|b|$, the claim is obvious. Now let $j \geq 1$; we assume that r_j can be written as $r_j = s_j \cdot a + t_j \cdot b$ and r_{j-1} can be written as $r_{j-1} = s_{j-1} \cdot a + t_{j-1} \cdot b$. (This is simply the induction hypothesis.) We must show that r_{j+1} also has such a representation. Indeed, we see that

$$\begin{aligned}r_{j+1} = r_{j-1} - q_j \cdot r_j &= s_{j-1} \cdot a + t_{j-1} \cdot b - q_j \cdot s_j \cdot a - q_j \cdot t_j \cdot b\\ &= (s_{j-1} - q_j s_j) \cdot a + (t_{j-1} - q_j t_j) \cdot b.\end{aligned}$$

So we can set $s_{j+1} = s_{j-1} - q_j s_j$ and $t_{j+1} = t_{j-1} - q_j t_j$, which means that the induction is complete. ■

Bézout's Lemma is extremely useful for proving properties of the gcd. As an application, we derive a fundamental characterization of numbers that are coprime to each other.

1.4.3. Corollary.
Two numbers $a, b \in \mathbb{Z}$ are coprime if and only if there are $s, t \in \mathbb{Z}$ such that $s \cdot a + t \cdot b = 1$.

Proof. By definition, a and b are coprime if and only if $\gcd(a, b) = 1$. If $\gcd(a, b) = 1$, then the numbers s and t exist by Bézout's Lemma.

Conversely, suppose there are numbers s and t with $s \cdot a + t \cdot b = 1$. Set $k := \gcd(a, b)$; we need to show that $k = 1$. Since k divides a and also b, we see that $k \mid s \cdot a + t \cdot b$ and hence $k \mid 1$. (Recall Theorem 1.2.2.) But 1 and -1 are the only integer divisors of 1, and k is not negative by definition. So we must have $k = 1$, as claimed. ■

Exercises.

1.4.4. Exercise. Determine $\gcd(135, 36)$ and $\gcd(851, 1449)$, using the Euclidean algorithm. For each of these pairs, also determine a representation of the gcd according to Bézout's Lemma.

1.4.5. Exercise. Use the Euclidean algorithm to compute $\gcd(1961, 1591)$. Compare this with the solution to Exercise 1.3.8 – which method is easier?

1.4.6. Exercise. Let a and b be integers with $\gcd(a, b) = 1$. Compute $\gcd(a^2 - b^2, a + b)$ and $\gcd(a^2 + b^2, a + b)$.

1.4.7. Exercise (P). Implement the Euclidean algorithm in a common programming language. Use this program to compute the greatest common divisor of $a = 1\,726\,374\,899\,084\,624\,209$ and $b = 6\,641\,819\,896\,288\,796\,729$. (Most languages support numbers of this size directly, but you need to make sure to use 64-bit integers.) Also write a program that finds the numbers s and t from Bézout's Lemma.

Further Exercises and Comments.

1.4.8. The Euclidean algorithm was described by the Greek mathematician and philosopher Euclid around the end of the fourth century BC in the seventh book of his *Elements* (a collection of 13 books). It is very likely that the algorithm was also known to earlier Greek mathematicians.

1.4.9. Exercise. Let a, b, and m be integers. Show, without using the Fundamental Theorem of Arithmetic, that $\gcd(am, bm) = \gcd(a, b) \cdot m$.

Use this observation to give a different proof of Theorem 1.3.6, and hence of the Fundamental Theorem of Arithmetic.

(*Hint:* For the first part, apply the Euclidean algorithm to a and b and also to am and bm, and compare these two applications.)

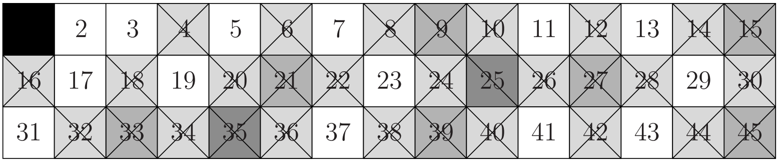
1.4.10. Exercise. Prove Bézout's Lemma without using the Euclidean algorithm. (*Hint:* Consider the set M of natural numbers that can be written as $ra + qb$ with $r, q \in \mathbb{Z}$. By the well-ordering principle, M has a smallest element k. Show that $M = \{nk : n \in \mathbb{N}\}$ and conclude that $k = \gcd(a, b)$.)

Note that this proof gives yet another characterization of the gcd!

1.5. The Sieve of Eratosthenes

One of the simplest ways to test a number for primality is the **Sieve of Eratosthenes**. Given a number N, this method finds all prime numbers $p \leq N$. The idea is to "sieve out" all composite numbers, step by step, until only primes are left. To begin, write down a list of all numbers from 1 to N. Then proceed as follows (see Figure 1.2):

- Since 1 is not prime, delete (cross out) 1 and begin with 2.
- 2 must be prime since the only possible divisors are 1 and 2. Cross out all other multiples of 2, beginning with 4, since they cannot be prime.
- The number 3 has not been deleted, so it must be prime. Cross out all other multiples of 3, beginning with 6.
- The next remaining number is 5. Cross out its multiples from 10 onwards and continue with the next prime, until we have arrived at the number $\lceil\sqrt{N}\rceil$ (i.e. the smallest number whose square is at least N). Numbers between $\sqrt{N}$ and N that are not prime must have at least one factor that is at most as big as $\sqrt{N}$ (Exercise 1.2.6) and hence have already been crossed out. Therefore we can stop; the remaining numbers are exactly the primes between 1 and N.



■	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~	11	~~12~~	13	~~14~~	~~15~~
~~16~~	17	~~18~~	19	~~20~~	~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	29	~~30~~
31	~~32~~	~~33~~	~~34~~	~~35~~	~~36~~	37	~~38~~	~~39~~	~~40~~	41	~~42~~	43	~~44~~	~~45~~

Figure 1.2. Applying the Sieve of Eratosthenes to the numbers up to 45, we obtain the primes 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, and 43. (The different shades indicate in which step of the procedure numbers were removed from the list.)

This method can easily be implemented on a computer. However, for numbers with large numbers of digits it cannot be used in practice because the procedure is **inefficient**; see Comment 1.5.3. (The precise meaning of this term is discussed in Section 2.3.)

Exercises.

1.5.1. Exercise. Use the Sieve of Eratosthenes to find all primes p with $p \leq 200$.

Further Exercises and Comments.

1.5.2. Eratosthenes of Cyrene was a librarian at the fabled *Library of Alexandria* in the third century BC. He is also known for his (surprisingly accurate) determination of the earth's circumference.

1.5.3. To use the Sieve of Eratosthenes for the one-hundred-digit number $N = 10^{100}$, we would have to make a list of N numbers. However, N is considerably larger than current estimates for the number of atoms in the universe.

Furthermore, we would consider each number from 1 to N at least once. Let us suppose that we can carry out 10^{20} such operations in a second – an optimistic assumption since this exceeds the abilities of current supercomputers by several orders of magnitude. Then we would still need 10^{80} seconds to finish using the Sieve of Eratosthenes. By current estimates, the universe is around 14 billion years old – this is less than 10^{18} seconds! For the purposes of cryptography, where numbers with several thousand digits are routinely used, this number N is not even particularly large . . . .

1.6. There are infinitely many primes

When performing the Sieve of Eratosthenes for a large number N (as in Exercise 1.5.1), we observe that, in the upper regions of the table, there are fewer and fewer prime numbers. This suggests a few questions: are there finitely many or infinitely many primes? How are they distributed? Are there regions where primes accumulate; are there gaps? Euclid already studied the first question. We now present his elegant proof that there are indeed infinitely many prime numbers.

1.6.1. Theorem.
There are infinitely many prime numbers.

Proof. We give a proof by contradiction; i.e. we assume that the claim is false. Then there are a number $n \in \mathbb{N}$ and primes $p_1, \ldots, p_n$ such that $\{p_1, ..., p_n\}$ is the set of *all* prime numbers. We consider the number $q := p_1 \cdot p_2 \cdots p_n + 1$. Then certainly $q \geq 2$, so there exists a prime p dividing q (by Exercise 1.2.6). Our assumption that $\{p_1, ..., p_n\}$ is the set of all primes forces $p \in \{p_1, \ldots, p_n\}$. Thus there is some $i \in \{1, ..., n\}$ such that $p_i \,|\, q$. However, the definition of q implies that q leaves remainder 1 when divided by p_i. This is a contradiction. ■

Although the set of primes is infinite, there are arbitrarily long pieces of the natural numbers where *no* primes can be found – so-called **prime gaps**.

1.6.2. Theorem (Prime gaps)**.**
Let $K \in \mathbb{N}$ be an arbitrary natural number. Then there exists a prime gap of length at least K. More precisely, the numbers $(K+1)! + 2$, $(K+1)! + 3, \ldots, (K+1)! + K + 1$ are all composite.

Proof. We may suppose that $K \geq 2$, as otherwise there is nothing to prove. Set $N := (K+1)!$. By definition, N is divisible by all numbers from 1 to $K+1$. So by Theorem 1.2.2, $N+2$ is divisible by 2, $N+3$

is divisible by 3, and in general $N+m$ is divisible by m for all m from 2 to $K+1$. Hence the numbers $N+2, N+3, \ldots, N+K+1$ are all composite. ■

When studying the distribution of primes, this is only the beginning: in Chapter 4 and in Appendix A, we will learn much more about this question, including precise theorems and conjectures. For example, even among very large numbers we can still find primes that are close together (see Comment 1.6.5).

Further Exercises and Comments.

1.6.3. Exercise. Show, in the proof of Theorem 1.6.2, that we could use the product of all primes less than or equal to K instead of $(K+1)!$ in order to construct a prime gap of length at least K. (Note that this is quite similar to the idea of the proof of Theorem 1.6.1.)

1.6.4. Our proof of Theorem 1.6.1 used an argument by contradiction. However, it is easy to rephrase it and to obtain a direct proof of the statement that, given any finite set of primes, one can always find another prime number that does not belong to this set.

1.6.5. It is believed that there are also infinitely many **twin primes**, i.e. pairs of primes that have distance 2 between them. Examples are 3 and 5, 5 and 7, or 1997 and 1999. Despite intensive research, this conjecture has still not been proved! We return to this question in Appendix A.

Further reading

An introduction to number theory, along with a discussion of the basic principles of mathematical proof, can be found in the books *An Introduction to Mathematical Reasoning* [**Ec**] and *Numbers, Groups and Codes* [**HP**]. For a more advanced, and far more detailed, discussion of the development of number systems, we refer the reader to *Numbers* by Ebbinghaus et al. [**Eb**]. A formal introduction to set theory and its connection to the development of the natural numbers can be found in the book *Naive Set Theory* by Halmos [**Hal**]. We treat further results from number theory in Chapter 3 and will give more references there.

Chapter 2

Algorithms and complexity

This chapter concerns **algorithms**: automated methods for solving problems. We begin by explaining the notion of an algorithm itself – in an informal manner that is sufficient for our purposes – and consider a number of examples. Then we explain the distinction between **decidable** and **undecidable** mathematical problems. The **complexity** of an algorithm is a measure of its "practicability". We use this notion to distinguish between **efficient** and **intractable** problems and discuss some ideas that can be used to improve the running time of algorithms. At the end of the chapter, we introduce **randomized algorithms**: methods that involve making a random choice (perhaps at the cost of a potential error in the outcome).

2.1. Algorithms

What is an algorithm? Since its beginnings, mathematics has included the search for methods that can be used to solve a given problem easily and essentially "automatically". Such procedures are called **algorithms**. The methods we learn in primary school to add, subtract, multiply, or divide numbers with many digits are all examples of algorithms, as are the Sieve of Eratosthenes and the Euclidean algorithm from the preceding chapter. With the development of electronic computers in the second half of the twentieth century, the search for algorithms and a systematic understanding of the underlying mathematics have gained an even greater importance.

But what is an algorithm? We imagine it as a set of instructions that we only need to follow, like a recipe, in order to solve the given problem. To explain this idea, let us examine two very different "algorithms". The first one actually is a recipe and the reader is invited to give it a try in his or her own kitchen.

Algorithm PANCAKE

Input: An egg, a small cup of flour, a small cup of milk, a pinch of salt, and a teaspoon of sunflower oil.

1. Mix the egg, flour, and milk in a bowl to make the pancake batter. Add the salt and mix again.
2. Heat the oil in a frying pan on medium heat until it sizzles.
3. Pour in the batter and softly shake the pan until the batter is distributed evenly.
4. Turn the pancake after 2–3 minutes.
5. After two more minutes, the pancake is ready to be eaten.

Our second "algorithm" tells you how to write a best-selling crime novel.

Algorithm CRIME-BESTSELLER

1. Invent a likable protagonist with some rough edges, ideally a private detective or police officer.
2. Also invent an (almost) perfect crime.
3. Design a story arc in which this criminal case is solved by the protagonist through investigation and possibly a sequence of lucky coincidences.
4. Write a novel that provides an entertaining and suspenseful account of this story.

It is immediately noticeable that these two sets of instructions are quite different from each other. While the first clearly describes the different steps and their order, the second one leaves many details open. Our examples may seem exaggerated, but they illustrate precisely the key property of an algorithm: its execution should not require us to think or to be creative. A correct performance of the steps described should always yield the same (correct) result, independently of the individual abilities of the person performing it.

We hope that the reader will agree with us that the pancake recipe qualifies as an algorithm in this sense (as far as a non-mathematical example possibly can), while the instructions for writing a crime novel do not even come close to matching our requirements.

Let us try to pinpoint the properties that we look for in an algorithm a little more precisely:

(a) it has a finite description;

(b) it consists only of "elementary" steps; in particular its realization does not require creativity or independent thinking;

(c) it is allowed to use arbitrarily large amounts of resources (paper, ink, memory, . . .), but this amount must be **finite** at any given time;

(d) at any time the next step to be carried out is uniquely determined by the results of the previous steps of the algorithm (**determinism**).

In other words, *an algorithm is a procedure that can – in principle – be implemented on a computer using a common programming language.* Of course we have not given a *definition* in the formal sense of mathematics. With some effort, it is possible to capture the notion of an algorithm mathematically (see below), but we will content ourselves with the above informal description and bring it to life with some examples.

Examples and explanations for the notion of an algorithm. We begin with a method that is usually taught in primary school: the addition of two numbers (with possibly many digits) on paper. We first remind the reader how this is normally done: the two numbers are written down, one above the other, so that the last digits are aligned. Then these last two digits are added together, with the last digit of the result being recorded below and the other digit (if there is one) “carried” over to the next step. Then the next-to-last digits are added, together with the “carried digit” if applicable, and so on, until all digits have been dealt with.

We now dissect this method and express it very precisely as an algorithm. For simplicity we phrase it for numbers that are written in **binary**, i.e. expressed in base two, rather than in the commonly used decimal system. (See Comment 2.1.4.)

Algorithm ADDITION

Input: Two natural numbers m and n, represented in binary as sequences of 0s and 1s.

1. Write down m above n in such a way that the two last digits of m and n are vertically aligned, the same for the next-to-last digits and so on. Draw a line underneath the two numbers.

2. If m and n have different numbers of digits, supplement the shorter of the two by leading zeros until both numbers have the same number s of digits.

3. Note down a zero in a separate “carry box” and set $j := 1$.

4. Let a be the j-th digit of m and b the j-th digit of n, *counted from the right.* Look in the table from Figure 2.1 that corresponds to the value in the carry box and read off the value in the a-th column and b-th row. We call this number k.

5. Write down the last digit of the number k in the j-th column (from the right) underneath the numbers m and n.

6. If k is a one-digit number, replace the number in the carry box by 0, otherwise by 1.

7. If $j \neq s$ (i.e. we have not arrived at the left), replace j by $j + 1$ and return to Step **4**.

8. Otherwise, write down the number in the carry box as the first digit of our result, ahead of the other digits we have computed. We are done.

a	0	1
b		
0	0	1
1	1	10

(a) Carry 0

a	0	1
b		
0	1	10
1	10	11

(b) Carry 1

Figure 2.1. Binary addition tables for the algorithm ADDITION.

The reader is invited to follow these instructions using an example of his or her choice. For the addition of 3 and 6 the various steps are shown in Figure 2.2. In the binary system, the number 3 is written as "11", and 6 is written as "110"; the result of our algorithm is "1001", which is indeed the binary representation of the number 9.

Although our description is somewhat awkward to read, this is nothing but the usual method of addition on paper. (Coincidentally, we chose to use binary representations instead of the decimal system only because this gives addition tables of a more manageable size.) Our set of instructions satisfies the criteria for an algorithm. Indeed, its description – consisting of the algorithm together with the addition tables from Figure 2.1 – is certainly finite. Each step contains an instruction that can be described with good conscience as being "elementary" and only uses a finite amount of resources. Finally, the order of steps and what is done in each step are uniquely determined, which means that the procedure is deterministic.

Perhaps we wish to add not just two numbers, but a whole list of these. We could try to formulate the usual written method for doing this, but there is an easier alternative.

Algorithm ADDITION-MANY

Input: A list of at least one and at most finitely many natural numbers, each represented in binary.

1. If the list contains only one number, we are done.
2. Otherwise apply the algorithm ADDITION to the last two numbers on the list, and replace these by the result of the addition.
3. Continue with Step **1**.

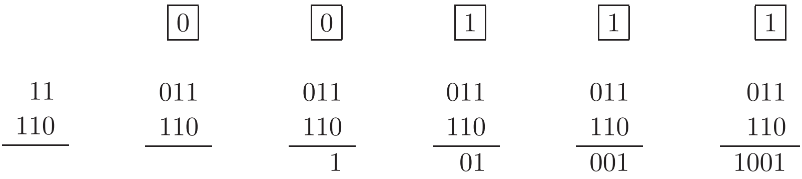


Figure 2.2. Addition of 3 and 6 using the algorithm ADDITION.

This illustrates an important property of algorithms: they can be combined. So when designing new algorithms, we should feel free to use others that we already know about. That simplifies life enormously – as we have seen in our first example, it can be quite an effort to decompose even simple methods into their most basic steps.

From now on, we shall allow basic operations on numbers (addition, multiplication, division) to be considered as elementary operations in our algorithms since there are well-known algorithms for these procedures (see Exercise 2.1.1). Indeed, these algorithms are implemented in every pocket calculator! Similarly, we frequently reuse algorithms that have appeared in earlier chapters or exercises.

Let us consider one more example. If we fix a number $n \in \mathbb{N}$, then it is known (see [**Lo**]) that $n \cdot \pi$ and $n \cdot e$ are not natural numbers, independent of the choice of n. So what about the number $n \cdot (\pi + e)$? One approach for finding a natural number n such that $n \cdot (\pi + e) \in \mathbb{N}$ is to use the following "algorithm":

```
Algorithm N*(PI+E)
1. Set $n := 1$.
2. Compute the number $a_n := n \cdot (\pi + e)$.
3. If the computed number $a_n$ is an integer, we are
   done.
4. Otherwise we replace $n$ by $n + 1$ and return to
   Step 2.
```

At first glance this looks like a legitimate algorithm. But if we take a closer look, we notice, for example, that nothing has been said about *how* to carry out the calculation in Step **2**. Of course we could use a calculator (or, in principle, we could do the calculation by hand if we are willing to spend a lot of time) to determine the number a_n up to any given precision, that is, rounded to a given number of decimal digits. But even if all of the digits we have computed are equal to zero, we still do not know for certain that $a_n \in \mathbb{N}$ because it could be that we simply did not compute enough digits. (As an example, the reader should use a pocket calculator to compute the number $a_{56602103}$.) Hence such a calculation is not sufficient to carry out Step **3** correctly. In view of these considerations we must conclude that the requirement that the steps in an algorithm be "elementary" is violated! Therefore N*(PI+E) is *not* an algorithm.

This example illustrates that we have to be a little bit careful when writing down algorithms. Nonetheless we hope to have convinced the reader that it should *always be possible to recognize whether a given procedure is an algorithm or not.* If, for every given step, it is clear that it can be carried out automatically and how to do so, then we are dealing with an algorithm; otherwise we are not. We explicitly encourage the reader to verify for every new algorithm presented in this book that the steps mentioned are either elementary or only require another algorithm that is already known.

We make a final remark regarding determinism, that is, part (d) of the conditions named at the beginning of the chapter. Certainly this requirement seems very sensible at first glance. However, there can be good reasons to weaken it slightly and to allow the algorithm to make **random choices**. We shall discuss the advantages of using such **randomized algorithms** in Section 2.5.

Formal definitions of the concept of an algorithm. The search for algorithms received a new significance at the end of the nineteenth and the beginning of the twentieth centuries. The first "real" computers would only appear in the middle of the twentieth century, but the mechanization of various processes during the industrial revolution had already sharpened the gaze of mathematicians for the algorithmic solution of problems.

David Hilbert, one of the leading mathematical thinkers of his time, proposed an ambitious program at the beginning of the twentieth century. He wished to provide mathematics once and for all with a formal foundation that would not contain any contradiction and in which every true mathematical statement could be proved. In particular, Hilbert was looking for a method (i.e. an algorithm) which could be used to determine the truth of any given mathematical statement. (This became known as **Hilbert's Entscheidungsproblem**[1].)

In the 1930s, Hilbert's questions motivated a number of mathematicians (including Alan Turing and Alonzo Church) to find a mathematical definition for the notion of an algorithm. Although the various concepts developed for this purpose look very different from each other at first, it was quickly realized that they are all equivalent. The same is true of all other definitions of algorithms that have been proposed since then. In particular, the procedures that can be implemented in one of today's common computer programming languages are exactly those that can be described in Turing's machine model. For this reason, it is believed that any and all of these definitions really do capture our intuitive notion of an "algorithm". This is called the "Church-Turing thesis" and justifies our decision to use only an informal definition of algorithms throughout this book.

Exercises.

2.1.1. Exercise. Formulate algorithms for

(a) the multiplication of two natural numbers,
(b) the subtraction of one natural number from another, and
(c) the division of a natural number by another with remainder.

Also formulate an algorithm that determines whether the first of two given natural numbers is larger than the second.

2.1.2. Exercise. Verify that the "Euclidean algorithm" from Section 1.4 really satisfies our requirements for an algorithm.

Further Exercises and Comments.

2.1.3. The term "algorithm" is derived from the name of the Persian mathematician *al-Khwarizmi*, the author of an important treatise on algebra in the ninth century AD.

[1] "Entscheidungsproblem" is German for "decision problem".

2.1.4. The **binary system**, which our readers might have encountered in school, uses only the digits zero and one. (In contrast, the commonly used **decimal system** uses the digits from zero to nine.) For example, the numbers from zero to ten are written as follows in the binary system: 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010.

Internally, computers always represent numbers in the binary system. The reason is that memory cells usually have exactly two states and that, therefore, each such cell can contain exactly one binary digit (zero or one).

2.1.5. A relatively simple proof that π is not a rational number can be found in [**NZM**]. The exercises of [**NZM**] also contain a proof of the fact that the Euler constant e is irrational. On the other hand, the question underlying our "algorithm" N*(PI+E), namely whether $\pi + e$ is irrational, is still open!

An even more difficult task is to distinguish among irrational numbers between the so-called "algebraic" numbers such as $\sqrt{2}$ and $\sqrt[4]{3}$ and **transcendental numbers** like e and π. For a discussion of these concepts, we refer to Chapter 2 of [**Lo**].

2.1.6. Exercise (P). We consider the following simple algorithm:

```
                    Algorithm COLLATZ
Input:  A natural number $n$.
  1. Write down the number $n$.
  2. If $n = 1$, we are done.
  3. If $n$ is even, replace $n$ by $\frac{n}{2}$. Otherwise, replace
     $n$ by $3n + 1$.
  4. Return to Step 1.
```

(a) Implement the algorithm in a common programming language.

(b) Perform the algorithm for the numbers 1 to 100. What do you notice?

It is reasonable to expect that the algorithm COLLATZ always calculates the number 1 at some point and then stops. This is called the $3n+1$**-problem** or also the **Collatz Conjecture** (after the mathematician Lothar Collatz who formulated this problem in 1936). Even though it has been verified for more than the first 5 000 000 000 000 000 000 natural numbers using computer experiments, no proof of the Collatz Conjecture is known!

2.2. Decidable and undecidable problems

The purpose of algorithms is to automatically solve **problems**, so we should clarify what we mean by a “problem”. Again, we shall content ourselves with an informal concept: to describe a problem, we specify a collection of **instances** or **inputs**, and for every such instance we also identify at least one correct **output**. Often (but not always) there is only one such output for each given input. In this case, the problem can be concisely stated as a question, for example as follows:

Problem SUM

Input: Two natural numbers n and m.
Question: What is $n + m$?

Inputs and outputs are usually represented as **strings**, i.e. as finite sequences of letters taken from some given alphabet. To maintain the informal character of our discussion, we usually refrain from discussing in detail how the instances should be coded as strings. When the inputs are natural numbers (which is the case for those problems we are most interested in), we always assume them to be represented in the binary or decimal system.

We can now say that an algorithm **solves** a problem if, for any possible input, the execution of the algorithm ends after finitely many steps with a correct output. A simple example is the algorithm ADDITION from the previous section, which solves the problem SUM. A problem that can be solved by an algorithm is called **decidable** (or **computable**); otherwise the problem is **undecidable**.

We also distinguish **decision problems**, where the desired output is always either “yes” or “no”, from the more general notion of **search problems**. For example, the central problem of this book is a decision problem:

Problem PRIMES

Input: A natural number $n \geq 2$.
Question: Is n prime?

In the next section we will discover – perhaps surprisingly – that every search problem can be reduced to an equivalent decision problem. Therefore we can restrict ourselves to the study of decision problems whenever this simplifies our discussions.

A decision problem has two different types of inputs. For **positive** instances, the desired answer is "yes", while it is "no" for the remaining, **negative** instances. For the problem PRIMES, each prime number is a positive instance, while the negative instances are precisely the composite numbers.

For every decision problem there is the so-called **dual** problem that is obtained by switching the roles of positive and negative instances. For PRIMES, the dual problem is compositeness:

Problem COMPOSITES

Input: A natural number $n \geq 2$.

Question: Is n composite?

It may not seem reasonable to differentiate between a decision problem and its dual problem: if one of them is decidable, then the other one is as well – we need only exchange the answers "yes" and "no". However, in Sections 2.4 and 2.5 we will encounter concepts in which positive and negative instances play different roles and where, hence, this distinction is important!

Finally we remark that problems with only *finitely many* different inputs are automatically decidable by an algorithm. Indeed, for such a problem it is possible to make a finite list that contains a correct output for each of the finitely many different inputs. Then our algorithm only has to check this list to solve the problem. We emphasize that the definition of decidability only requires the *existence* of an algorithmic solution. To prove that a problem is computable, we do not necessarily have to write down an explicit algorithm but could instead argue by contradiction or by distinguishing several cases (see also Comment 2.2.4).

For example, consider the problem "Is the number $m := 2^{2^{61}-1}-1$ prime?". It is decidable because, if m is prime, then it is solved by the algorithm that always outputs “yes”, and otherwise, it is solved by the algorithm that always answers “no”. Which one of the two is right is an entirely different question! That being said, we will give explicit algorithms for all decidable problems encountered in this book.

Undecidable problems. As already mentioned, in the early twentieth century Turing and Church undertook formal investigations into the notion of an algorithm. In the end, these led to the realization that Hilbert’s dream of a solution of the Entscheidungsproblem cannot be attained. Indeed, Turing observed that algorithms (which, after all, have a finite description, e.g. in the form of their source code when implemented in a given programming language) can again be used as inputs. He thus considered the following decision problem:

Halting problem

Input: An algorithm A and an input x for A.

Question: Does the algorithm *halt* on the input x? I.e. does its execution end after finitely many steps?

2.2.1. Theorem (Undecidability of the halting problem)**.**
There is no algorithm that solves the halting problem.

Proof. Assume, by contradiction, that there is an algorithm HALT that solves the halting problem. Consider the following algorithm:

```
Algorithm DIAG

Input:  An algorithm $B$.
  1. Execute HALT with $A = B$ and $x = B$.
  2. If the output in Step 1 was “no”, we are done.
  3. Otherwise repeat Step 2.
```

In other words, DIAG will halt on input B if and only if B does *not* halt when given its own source code as input. In particular, algorithm DIAG will halt when given its own source code as input if and only if it does *not* halt when given its own source code as input. That is impossible! ■

The undecidability of the halting problem has a number of important consequences.

- Not every problem can be solved by an algorithm. When faced with a question whose solution we would like to automate, we must thus keep in mind that such an automated procedure might not exist (see Comment 2.2.3); and when it does, finding it may not be easy at all . . . .
- In particular, Hilbert's Entscheidungsproblem has no solution: there is no program that can test whether a mathematical statement is true or false. Indeed, otherwise we could also use this method to solve the halting problem! Hence a computer is not a panacea for mathematical problems – it can require a lot of work and inventiveness to use it successfully in mathematical research.
- There is no automatic procedure to test whether a given program is written correctly, not even to detect whether it ever enters a never-ending loop and hence does not halt. There are very short and simple algorithms for which this is not known; recall Exercise 2.1.6. In other words, programmers must pay close attention to make sure that the programs they are writing really do what they are supposed to do. Source code needs to be written clearly, with many comments, because an automatic check of a badly written program is not possible.

Exercises.

2.2.2. Exercise. Which of the following problems are decidable, i.e. can be solved by an algorithm?

(a) Let n and k be two natural numbers. Is k a divisor of n?

(b) Is the Riemann Hypothesis true? (The Riemann Hypothesis is a famous open mathematical problem; see Section 4.3.)

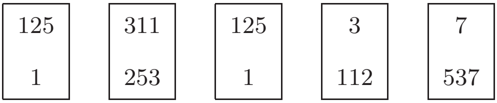


Figure 2.3. Dominos as in the Post Correspondence Problem (Comment 2.2.3(a)).

(c) Given a computer program, decide whether there is an input that causes the program to output the number 42.

(d) Let $A \subseteq \mathbb{N}$ be a finite set. Is it possible to subdivide A into two sets X_1 and X_2 in such a way that the sum of the elements in X_1 is the same as the sum of the elements in X_2?

(e) Suppose that we have a list of finitely many natural numbers. Order these numbers according to their size.

Further Exercises and Comments.

2.2.3. There are many problems that are known to be undecidable. Among them are some that might surprise you! Here are two famous examples.

(a) **The Post Correspondence Problem.** As input, we are given a finite set of dominos (see Figure 2.3). On each of these, two natural numbers have been engraved (in decimal notation): one on the top half and another (possibly different) number on the bottom half.

Suppose that we can make arbitrarily many copies of each of these dominos. Is it possible to place finitely many such pieces next to each other in such a way that the sequence of digits in the upper row is the same as the sequence of digits in the lower row? (In the example from Figure 2.3 this is possible; the sequence of digits is 12531112537.)

(b) **Hilbert's Tenth Problem.** Let $P = P(X_1, \ldots, X_n)$ be a **polynomial** in the n variables $X_1, \ldots, X_n$ and with integer coefficients. (This means that P is a finite sum in which every term has the form $C \cdot X_1^{k_1} \cdots X_n^{k_n}$, where $C \in \mathbb{Z}$ and $k_1, \ldots, k_n \in \mathbb{N}_0$; see also Section 2.5.) Does P have an integer root? That is, are there numbers $x_1, \ldots, x_n \in \mathbb{Z}$ such that $P(x_1, \ldots, x_n) = 0$?

2.2.4. As we remarked previously, it might be possible to prove the decidability of a problem without explicitly exhibiting an algorithm. There are indeed cases where this has happened! One example is the problem of deciding whether a given **graph** is "knot-free". (I.e. is there a way of drawing the graph in three-dimensional space without creating any knots that can be untangled only with scissors? For background and definitions from graph theory, we refer the interested reader to the book *Graph Theory* [**Dst**] by Diestel.)

For a long time it was unknown whether this problem is decidable or not. However, at the beginning of the twenty-first century, the famous "graph minor theorem" was proved. It implies the existence of an algorithm that decides whether a graph is knot-free or not – even an **efficient** algorithm in the sense of the next section. However, no explicit example of such an algorithm has yet been found!

2.2.5. Exercise. In addition to the solution of his Entscheidungsproblem, Hilbert desired the introduction of a formal system in which a mathematical statement is true if and only if it can be proved. Argue informally that this would imply the computability of the halting problem and is therefore impossible. (*Hint:* The correctness of a mathematical proof, suitably formalized, can be checked algorithmically.)

Before Turing's work it had already been proved by **Kurt Gödel** that there is no such system. The discovery that there are statements that can be neither proved nor disproved using the usual mathematical axioms shocked the mathematical community. This idea and much more is explored in the book *Gödel, Escher, Bach* [**Ho**], which is highly recommended. For an informal but extensive discussion of Gödel's incompleteness theorem mentioned above we refer to the book *Gödel's Theorem: An Incomplete Guide to Its Use and Abuse* [**Frz**] and to the book *Mathematical Logic* [**EFT**] for a formal treatment of propositional logic.

2.2.6. The method used in the proof of Theorem 2.2.1 is referred to as **diagonalization**. It is frequently used in mathematics to derive contradictions in situations where there is some form of *self-reference*. (In our case this self-reference comes from the fact that its own source code can be given as input to an algorithm.) Using this method, one can prove, for example, that there is no *set of all sets* and that the real numbers are *uncountable*. (This means that it is not possible to write down all real numbers as a simple infinite sequence. On the other hand, this *is* possible for the integers and even for the rational numbers.) Gödel used diagonalization to prove the incompleteness theorem mentioned above.

2.3. Complexity of algorithms and the class P

As seen in the last section, it can be difficult or even impossible to find an algorithm that solves a given mathematical problem. But this is not the end of our troubles! We also need to consider a method's

efficiency, i.e. whether or not it is possible to perform this procedure in practice. Indeed, an algorithm whose memory requirements are larger than the number of atoms on earth or whose execution takes several billion years would not be very helpful.

Thus we have to concern ourselves with the **resources** required by an algorithm. The manner in which these depend on the input is called the **complexity** of the algorithm and its study is known as **complexity theory**. We shall restrict ourselves to the temporal aspect of complexity – how long does it take to carry out the given procedure? – because this tends to be most important, both in theory and in practice. If A is an algorithm and I is an input for A, then the **running time** of A on I is defined as the **number of elementary instructions** that are carried out when A is applied to I. The **running time function** of the algorithm associates to every $n \in \mathbb{N}$ the highest running time of A on an input of length n; this number is denoted by $s(n)$.

(We remind the reader that the inputs of an algorithm are *strings*; the "length" of an input is the length of this string, i.e. the number of characters. If the input is a natural number m in binary representation, its length is exactly $\lfloor \log(m) \rfloor + 1$.)

The precise value of the running time depends on what exactly we mean by an "elementary instruction". This in turn depends on the specific machine model used. For theoretical purposes, e.g. in Turing's model, only a very small number of elementary instructions are normally used, while in modern microprocessors a plethora of operations (e.g. elementary arithmetic for numbers up to a certain size) is already implemented on the hardware. The only important thing for us is that such an operation can be completed within a certain fixed amount of time, usually no more than the tiniest fraction of a second.

Asymptotic growth rates. We would like to evaluate the efficiency of an algorithm using its running time function and compare the complexity of different procedures. This may sound easy, as we can imagine computing the running time functions of both algorithms and decide which of them takes smaller values (on average, everywhere, for large numbers,...). However, keep in mind that the running time itself depends on the precise definition of an elementary operation. In view of rapid technological advances of computer technology, our considerations of efficiency should be *independent* of this definition. Furthermore, it is always possible to improve the performance of our algorithm for some inputs by considering suitable special cases separately. (For example, we could include the list of the first 1000, or even 1 000 000, primes in a primality test and hence considerably shorten the running time when one of these occurs as an input.) Hence a measure of efficiency that is meaningful beyond the current state of the art in computer technology ought to have the following properties:

- It should not distinguish between running time functions that differ at most by a multiplicative constant (since such a difference could result simply from a change in the machine model or implementation).
- It should restrict itself to conclusions about the behavior of the algorithm for "large" inputs.

There is a mathematical concept that satisfies precisely these requirements: the notion of the *asymptotic growth* of a function.

2.3.1. Definition (Asymptotic growth).
Let $f, g : \mathbb{N} \to \mathbb{R}$ be functions. We say that f *grows asymptotically at most as quickly as* g for $n \to \infty$, and write $f(n) = O(g(n))$, if there is a constant $C > 0$ such that

$$|f(n)| \leq C \cdot |g(n)|$$

for all sufficiently large n. (This last statement means that there is some $N \in \mathbb{N}$ such that the inequality is satisfied for all natural numbers $n \geq N$.)

Examples. We have $n = O(n^2)$, $n^2 = O(n^5)$, but $2^n \neq O(n^2)$ (see Exercise 2.3.4).

Remark. It is important to emphasize that the sign "=" in the O-notation does not indicate an actual equality. It might be more formally correct to consider $O(g)$ as a class of functions and to write

$f \in O(g)$. However, the above notation is well established and should not cause any confusion as long as we are careful.

Let us say, then, that a given algorithm has *asymptotic running time* at most $g(n)$ if its running time satisfies $s(n) = O(g(n))$. When studying the efficiency of algorithms, we discuss only this asymptotic running time. Hence an algorithm with $s(n) = 20000n$ will be considered to be more efficient than one that satisfies $s(n) = 5n^2$. (However, in practice it would be sensible to apply the second algorithm as long as the input is no longer than 4000 digits!)

Now we would like to separate "efficient" algorithms from those that really cannot be considered useful, and this separation should be independent of the current state of technology (see above). Computer scientists largely agree on the following definition for this purpose.

2.3.2. Definition (Efficient algorithms)**.**
An algorithm is called **efficient** if its running time function $s(n)$ is **polynomial**, i.e. $s(n) = O(n^k)$ for some positive number k.

The class of decision problems that are **efficiently computable**, i.e. for which an efficient algorithm exists, is denoted by **P**.

Problems for which there is no efficient algorithm are called **intractable**. The idea is that advances in technology can have a real impact on the execution of algorithms of polynomial running time, whereas this is impossible e.g. for an algorithm whose running time is exponential in the size of the input.

One might argue that a problem whose solution requires an asymptotic running time of $O\left(n^{10^{10^{10}}}\right)$ should not be considered to be tractable in practice (compare also Exercise 2.3.13). However, so far, the above separation has turned out to work rather well, in the sense that important problems that have been shown to belong to the class **P** tend to have algorithms that are indeed practical in real applications. In addition, we should think of the classification as giving us a *rough* division of problems. While important open questions remain even for this coarse classification (compare Section 2.4), it may not make sense to look for finer distinctions!

Examples. Let us begin by investigating the usual method of adding two numbers (Algorithm ADDITION). Here, for each digit, the algorithm adds two digits and possibly a carry. This yields the corresponding digit of the sum and the carry for the next digit.

So if our two numbers have k digits, we have to carry out an addition of at most three one-digit numbers at most k times. We can safely consider these to be elementary operations. *Hence the algorithm ADDITION has asymptotic running time $O(k)$.* In particular, it is efficient according to our definition, which is reassuring.

Naturally, the next algorithm we consider is the long multiplication of two natural numbers n_1 and n_2. This consists of two steps: first, n_1 is multiplied once by each digit of n_2, and the results are written one below the other, each shifted one digit from the next. Then these numbers are added. If n_1 and n_2 both have k digits, then, similarly as above, the multiplication of n_1 with a single-digit number has running time $O(k)$. Since we need to do this once for each of the k digits of n_2, we obtain an overall running time of $O(k^2)$ for the first part of our algorithm. The second part, i.e. the addition of k numbers, also requires a running time of $O(k^2)$.

In conclusion, the algorithm “long multiplication” has asymptotic running time $O(k^2)$. Hence this algorithm is also efficient, but not *as* efficient as addition. This agrees with our personal experience of using these methods!

Finally, we take a look at the ancient Sieve of Eratosthenes, which finds all prime numbers $p \leq n$ for a given number n and hence decides whether n is prime itself. This algorithm has the following properties:

(a) Every natural number from 2 to n is either recognized as prime or crossed out as a (composite) multiple of a prime.

(b) Every natural number from 1 to $\sqrt{n}$ is examined at least once to see whether it has already been crossed out.

(c) For every prime p from 2 to $\sqrt{n}$ all proper multiples of p, up to size n, are computed and crossed out. That is, we carry out $\left\lceil \frac{n}{p} \right\rceil - 1$ additions of p to some other number $k \leq n$.

(The reader is invited to work out the details more precisely.) From this analysis, we conclude: *the time required to determine all primes $\leq n$ using the Sieve of Eratosthenes is at least n and at most polynomial in n.*

Does this mean that the algorithm is efficient? If so, we have found an efficient algorithm for the problem PRIMES already and achieved the main aim of this book. However, we must remember that the running time of an efficient algorithm is *polynomial in the length of the input*, as a string, and here this means that it should be polynomial in $\log n$. So we realize that the method is highly inefficient since its running time is *exponential* in the length of the input. Anyone who has tried to run the Sieve for a 4-digit number, and compared this e.g. to the multiplication of two such numbers, will probably agree with our conclusion (also recall Comment 1.5.3.)

Divide and conquer. In the game "What's my line?", a popular television program running from the 1950s until the 1990s, the goal is to determine a guest's occupation using as few yes-or-no questions as possible. To be successful, it is clearly advisable to first determine a broad area of employment rather than asking very specifically from the beginning.

This simple idea also helps with the development of efficient algorithms. Let us not try to guess an occupation, but rather a number, say between 1 and 100. (That is not quite as much fun as the original game, but easier to study mathematically.) As with "What's my line?", we do not try all 100 numbers, one after the other, but instead reduce the possible choices by half in every step. For example, we could ask first whether the number is between 1 and 50. If so, we ask whether it is between 1 and 25, and so on. This way, we require at most $\lceil \log(100) \rceil = 7$ questions to find the answer, instead of up to 100 – a dramatic improvement!

Note that the underlying idea is to get a grip on a problem by dividing it into several pieces of essentially the same size and then studying these separately. This principle is known as **"divide and conquer"** and is extremely important in the theory of algorithms.

We now apply this idea to the problem of computing the power n^k of two natural numbers n and k.

The obvious manner of doing this would be to carry out $k-1$ multiplications (i.e. we compute n^2 by multiplying n with n, then multiply the result by n again, and so on). However, we can reduce the number of necessary multiplications considerably. For example, if $k = 2^j$ is a power of 2, we can write

$$n^k = n^{2^j} = \underbrace{\left(\dots\left(\left(n^2\right)^2\right)^2\dots\right)^2}_{j \text{ times}}.$$

In other words, to calculate n^k, we need to compute $n^{k/2} = n^{2^{j-1}}$ only *once* (instead of twice) and then carry out a single multiplication. So in the end, we require just $j = \log k$ multiplications.

The same idea also works when k is not a power of 2. Indeed, let $m = \lfloor k/2 \rfloor$; then we can write

$$n^k = \begin{cases} n^m \cdot n^m & \text{if } k \text{ is even,} \\ n \cdot n^m \cdot n^m & \text{if } k \text{ is odd.} \end{cases}$$

In either case, the number n^m appears twice in the formula but needs to be calculated only once. We thus obtain a *recursive* algorithm, meaning that the program may run itself, on a different input, at some point during the execution. (We invite the reader to think about how the algorithm could be formulated without recursion.)

```
POWER ALGORITHM
Input:  Numbers $n \in \mathbb{Z}$ and $k \in \mathbb{N}_0$.
  1. If $k = 0$, output "1".
  2. If $k = 1$, output "$n$".
  3. If $k \geq 2$, then write $m = \lfloor k/2 \rfloor$ and use the power
     algorithm to calculate $a = n^m$.
       (a) If $k$ is even, output "$a^2$".
       (b) If $k$ is odd, output "$n \cdot a^2$".
```

The key observation is that the power is reduced by a factor of 2 each time that Step **3** is called. It follows that this step is performed exactly $\lfloor \log k \rfloor$ times. Since this step requires at most two multiplications, we see that in total the power algorithm requires no more than $2\lfloor \log k \rfloor$ multiplications.

We emphasize that the power algorithm is not efficient in the sense of this section, simply because the *length* of the numbers involved in the multiplications quickly becomes large, and indeed the length of the output n^k will grow *exponentially* with $\log k$. Nonetheless, this method will turn out to be extremely important for the efficient algorithms that we design in the course of this book.

To conclude this section, we use "divide and conquer" to convert any search problem S into a corresponding decision problem D. For simplicity, we assume that the possible outputs for the problem S are natural numbers. It is not difficult to encode arbitrary strings using natural numbers – indeed this happens in all computer systems. (Furthermore, the most interesting problems for us are all of this form anyway.) We also suppose that for every input I of S, there is exactly one correct output, which we denote by $S(I)$. Many problems have this property; otherwise we can study the slightly stronger problem that asks for the *smallest* correct output. With these assumptions, let D be the following decision problem:

Problem D

Input: An instance I of S and a natural number k.

Question: Is $S(I) \leq k$?

2.3.3. Lemma (Equivalence of S and D).

Let S and D be as above. Suppose that there is a number $d \in \mathbb{N}$ with

$$\log \max_I S(I) = O(n^d),$$

where the maximum is taken over all instances I of length n. Then there is an efficient algorithm for S if and only if there is such an algorithm for D.

Remark. The hypothesis of the lemma means that the length of the desired output for B is at most *polynomial* in the length of the input.

Clearly a search problem which does not have this property cannot be solved by an efficient algorithm, as otherwise just *writing down* $S(I)$ will take too much time, to say nothing of its computation! (This is precisely the issue that we commented on when discussing the ineffiency of the power algorithm.)

Proof. If A is an algorithm that solves S efficiently, then we can also solve D efficiently by first running A with input I and then comparing the numbers $S(I)$ and k.

For the opposite direction, suppose that A is an efficient algorithm that solves D. If I is an instance of S, then we know by hypothesis that $S(I)$ belongs to the range from 1 to 2^{n^d}. So we can apply the strategy from the "What's my number?" game described above, with each question corresponding to one application of the algorithm A. For an instance of length n, we apply A at most $O(n^d)$ times to find $S(I)$ and hence have found an efficient algorithm for S! ■

Exercises.

2.3.4. Exercise (!). The goal of this exercise is to establish the following facts, which should be intuitively clear: adding a constant to a function does not change its order of growth, polynomials of the same degree have the same order of growth while higher-degree polynomials have a higher order of growth, exponential functions grow faster than any polynomial, and the logarithm grows slower than any polynomial.

(a) Let $f : \mathbb{N} \to \mathbb{R}$ be a function that is bounded below by a positive constant. (That is, there is a number $\varepsilon > 0$ such that $f(n) > \varepsilon$ for all $n \in \mathbb{N}$.) Show that $f(n) + C = O(f(n))$, where $C \in \mathbb{R}$ is an arbitrary constant.

(b) Let $k, m \in \mathbb{N}_0$. Show: $x^k = O(x^m)$ if and only if $k \leq m$.

(c) Let P be an arbitrary polynomial of degree at most d. Show that $P(n) = O(n^d)$.

(d) Let $a > 1$ be a real number. Is it true that $a^n = O(2^n)$?

(e) Let $\varepsilon > 0$ be a real number. Show that $\log n = O(n^\varepsilon)$.

(f) Let $k \in \mathbb{N}$. Show that $n^k = O(2^n)$.

(*Hint:* For (e), use the power rule $\log n^\varepsilon = \varepsilon \log(n)$ and Exercise 1.1.12(a). Alternatively, derive (e) from (f). For (f), show first that $(n+1)^k < 2n^k$ for all sufficiently large $n \in \mathbb{N}$. Deduce from this, using mathematical induction, that there is a constant $C \in \mathbb{R}$ such that $n^k \leq C \cdot 2^n$ for all sufficiently large numbers $n \in \mathbb{N}$.)

2.3.5. Exercise (!).

(a) Check that long division with remainder is an efficient algorithm.

(b) Let $m, n \in \mathbb{N}$ with $m > n$. Show that the number of long divisions that are used in the Euclidean algorithm to find $\gcd(m, n)$ is of magnitude $O(\log m)$. (In other words, the algorithm is efficient.)

Hint: Convince yourself that, after any two steps, the larger of the two numbers under consideration has decreased at least by half.

2.3.6. Exercise (!). Find efficient algorithms for the following problems:

(a) Given $n, k \in \mathbb{N}$, compute the number $\lfloor \sqrt[k]{n} \rfloor$ (i.e. the largest natural number m with $m^k \leq n$).

(b) Given $n \in \mathbb{N}$, is n a perfect power? I.e. are there natural numbers m and $k > 1$ such that $n = m^k$?

2.3.7. Exercise. Formulate an equivalent decision problem for each of the following search problems:

(a) Input: a natural number n. Output: a non-trivial divisor of n, if such a divisor exists.

(b) Input: a natural number n. Output: the number of divisors of n.

Further Exercises and Comments.

2.3.8. The “big-O notation” that we use to study asymptotic running times was introduced at the end of the nineteenth century by the number theorist Paul Bachmann. It was popularized by the mathematician Edmund Landau and is today associated mainly with him (“Landau symbols”).

2.3.9. The term “divide and conquer” (*divide et impera*) is often connected with Julius Caesar, although its exact origin is unclear. It originally describes the method of generating distrust and discord among opponents in order to defeat and rule them more easily.

2.3.10. Exercise. Let $f_1, f_2, \ldots$ be the Fibonacci numbers, as defined in Section 1.1. The definition suggests the following recursive algorithm for computing f_n:

Algorithm FIB

Input: A natural number n.

1. If $n = 1$ or $n = 2$, then $f_n = 1$ and we are done.

2. Otherwise first compute f_{n-1} and then f_{n-2}, using the algorithm FIB.

3. The desired result is the sum of the two numbers computed in Step **2**.

How many operations does the computation of f_n require when using this algorithm? How can we reduce the number of operations?

2.3.11. Exercise. We study the running time of the Euclidean algorithm more closely than in Exercise 2.3.5. Let $a, b \in \mathbb{N}$ and let $r_0, \ldots, r_{j_*}$ be the numbers that occur when the algorithm is run with inputs a and b. Recall that $r_{j_*} = 0$ and $r_{j_*-1} = \gcd(a, b)$. For simplicity, let us assume that $a > b$.

(a) Let f_1, f_2, $\ldots$ again denote the sequence of Fibonacci numbers. Prove the following: if $k \in \mathbb{N}$ with $k \leq j_*$, then $r_{j_*-k} \geq f_{k+1}$.

(b) Deduce that the Euclidean algorithm carries out at most $k - 2$ divisions with remainder, where k is the largest natural number that satisfies $f_k \leq a$.

(c) Let $k \geq 3$. Prove that, when running the Euclidean algorithm with $a = f_k$ and $b = f_{k-1}$, exactly $k - 2$ divisions with remainder are required.

2.3.12. Exercise. As we saw, the usual long multiplication on paper has a running time of $O(n^2)$. However, it is possible to use other multiplication methods that reduce the running time. A simple example is the **Karatsuba algorithm**, which is based on the principle "divide and conquer" and which we develop in this exercise.

(a) Let $a = a_1 + a_2 \cdot 2^j$ and $b = b_1 + b_2 \cdot 2^j$ be natural numbers. Show that $a \cdot b = a_1 \cdot b_1 + a_2 \cdot b_2 \cdot 2^{2j} + (a_1 \cdot b_1 + a_2 \cdot b_2 - (a_1 - a_2) \cdot (b_1 - b_2)) \cdot 2^j$. (In this formula there are only *three* (not four) *different* products of smaller numbers.)

(b) Use this idea to formulate a recursive algorithm for the multiplication of two numbers a and b of length k (this is the Karatsuba algorithm). This algorithm should first subdivide both a and b into their first $k/2$ digits and their last $k/2$ digits and then carry out three multiplications of numbers with $k/2$ digits.

(c) Show that the asymptotic running time of this algorithm is

$O(3^{\log k}) = O(k^{\log 3})$. We have $\log 3 \approx 1.6$; so the Karatsuba algorithm is (asymptotically) more efficient than long multiplication. However, in practice there will only be an improvement for numbers with several hundred digits because the constants in the Landau notation turn out to be very large.

The time complexity of multiplication can be reduced even further: the **Schönhage-Strassen algorithm** has running time $O(k \cdot \log(k) \cdot \log\log(k))$. In practice this algorithm is only used for numbers with tens of thousands of digits. An algorithm with even smaller asymptotic running time was presented by Fürer in 2007 [**Fü**].

2.3.13. Exercise. Let $m, n \in \mathbb{N}_0$. We define a function (the **Ackermann function**) by

$$A(m,n) := \begin{cases} n+1 & \text{if } m = 0, \\ A(m-1, 1) & \text{if } m > 0 \text{ and } n = 0, \\ A(m-1, A(m, n-1)) & \text{otherwise.} \end{cases}$$

(a) Find explicit formulae for $A(m,n)$ when $0 \le m \le 3$. Then show

$$A(4,n) = \underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{n+3 \text{ times}} -3.$$

(b) Let $n \in \mathbb{N}_0$. The **inverse Ackermann function** is defined as

$$\alpha(n) := \min\{k \in \mathbb{N} : A(k,k) \ge n\}.$$

Show that $\alpha(n) = O(\log n)$.

(c) In fact the function $\alpha(n)$ grows far slower than $\log n$. For example, we have $A(4,4) \gg A(4,2) = 2^{65536} - 3$. This number is several orders of magnitude higher than today's estimates on the number of atoms in the entire universe. So for practical purposes, we always have $\alpha(n) \le 4$. Is an algorithm with running time $O(n^{\alpha(n)})$ more efficient than one with running time $O(n^5)$ according to our definitions? Is such an algorithm considered efficient at all?

2.4. The class NP

In the previous section, we introduced the class **P** of efficiently solvable problems. Now we briefly discuss the equally important class **NP** of **efficiently verifiable** problems: informally speaking, for these a solution can be *verified* in polynomial time. This definition allows us to discuss one of the central problems of complexity theory and of mathematics in general: the $\mathbf{P} \overset{?}{=} \mathbf{NP}$ question. The idea of a **witness** that is used in the definition of **NP** will also be useful when studying randomized algorithms in the next section.

As motivation, take a look at the problem COMPOSITES. To show that a number n is a positive instance, we "just" have to pull a non-trivial divisor k of n out of our hat – then we can easily check, using long division, that k really does divide n. We call k a **witness** for the compositeness of n. Let us make this a bit more precise: we have a decision problem D (in our example this is the problem COMPOSITES). In addition, we have a set P of "possible witnesses" (in our example these are all natural numbers $k \geq 2$) and another decision problem E (the question "Is k a non-trivial divisor of n?") with the following properties:

(a) The inputs for E are pairs of the form (I, p), where I is an instance of D and $p \in P$.

(b) E can be solved efficiently.

(c) I is a positive instance of D if and only if there is at least one $p \in P$ such that (I, p) is a positive instance of E.

(In our example, the final claim says that a number is composite if and only if it has a non-trivial divisor.) In addition, it is important that every positive instance I of D has a witness that is "not much bigger" than I; i.e. there are constants C and m such that the following holds:

(d) If I is a positive instance of B of length n, then there is a positive instance (I, p) of E such that the length of p is at most $C \cdot n^m$.

2.4.1. Definition (Class **NP**).
If D is a decision problem for which there exist a set P and a decision problem E with the properties (a) to (d), then we say that D is **efficiently verifiable**. If (I, p) is a positive instance of E, then we call p a **witness** for I. The class of all efficiently verifiable problems is denoted by **NP**.

We remark that every problem D that belongs to the class **P** also belongs to **NP**. Indeed, in this case it does not matter which set of witnesses we use, and E is simply the problem "Given (I, p), is I a positive instance of D?". Hence we see that $\mathbf{P} \subseteq \mathbf{NP}$.

Examples. To illustrate the definition further, let us investigate which of the following problems belong to the class **NP**.

(a) We are given a map (e.g. as a finite list of country names, together with a table that tells us which countries are neighbors). Is it possible to color the countries with the two colors red and green in such a way that neighboring countries never have the same color?

(b) The same problem as in (a), but with three colors, e.g. red, blue, and green.

(c) We are given a map and a number k. Are there at least k different possibilities to color the map as in (b)?

(d) We are given a list of students, teachers, and subjects. For each student, we know which subjects he or she wants to choose and, moreover, we know which subjects each teacher is capable of teaching. With this information, is it possible to design a weekly schedule that takes into account all these constraints?

The first problem is easy to solve. We begin by coloring the first country on the map green. Then all its neighbors must be colored red, all their neighbors green, and so on. If we encounter a country that has already been colored, then there are two possibilities: if it already has the color that we were going to use for it, then everything is fine and we continue our algorithm. Otherwise we have seen that the given map cannot be colored with two colors, and we are done!

Clearly this algorithm only needs to consider each country and each border once to either obtain the desired coloring or to realize that this is impossible. Hence the *two-coloring* problem can be solved efficiently, so it belongs both to the class **P** and in particular to **NP**.

Now we extend the problem to three colors and see that it becomes more difficult: we can still try to color the map by assigning each country a color that is admissible at this step. However, if we run into a problem, this does not prove that the map cannot be colored at all. After all, in each of the previous steps we had a *choice* of colors, and it could be that we simply made a wrong choice somewhere along the way.

Hence it is not clear whether the problem belongs to **P**. However, it definitely belongs to the class **NP**. A possible witness is simply a map together with a coloring with three colors. To check whether a coloring satisfies our conditions, we simply have to go through the list and verify that neighboring countries never have the same color. This is an efficient algorithm and therefore the "three-coloring" problem is efficiently verifiable.

In (c) we are asking for the *number* of such colorings. Here it may also seem, at first glance, that there is a simple witness: to show that there are at least k colorings, surely we only have to give these k colorings?

But we have fallen into a trap. The issue is the same as when we studied the Sieve of Eratosthenes in the previous section. An instance of our original problem consists of a map and the number k, in binary representation. A list of k different colorings hence requires, in general, an amount of memory that is *exponential* in the size of the input. Hence the problem probably does not belong to **NP**.

Finally, checking whether a given schedule of lessons is correct is easy: the last problem belongs to **NP**.

An interesting property of the class **NP** is that positive and negative instances play extremely different roles. Indeed, *negative* instances of such a problem are characterized by the fact that they have no witness. *Thus, if D belongs to* **NP**, *the dual problem might not belong to* **NP**! As an example we again look at the problem COMPOSITES, which we know to belong to **NP**. The dual problem is PRIMES, i.e. the question whether n is prime. While a witness for COMPOSITES can be obtained directly from the definition of primality, we are currently empty-handed when it comes to PRIMES.

On the other hand, the fact that we do not see a witness at first glance does not prove that it does not exist! After all, it is the goal of this book to find an efficient algorithm for PRIMES and hence to prove that PRIMES belongs to the class **P** and thus also to **NP**. (The proof of PRIMES $\in$ **NP** is much easier than that of PRIMES $\in$ **P** and was already found in 1975 by Vaughan Pratt; see for example [**P**, Section 10.2].)

There is one issue we have not mentioned: are there problems that belong to **NP** but not to **P**? In other words, is it generally more difficult to find the solution to a problem than to check such a solution? We would expect this to be true!

Indeed, many problems in **NP** are believed to be intractable. The above-mentioned problems of three-coloring and creating a class schedule are of this type. But proving that one of these problems does not belong to **P** presents great difficulty: there are infinitely many possible algorithms, and one of these might use a non-obvious "trick" to find the desired solution. How are we going to exclude this possibility? So far, nobody has been able to solve this question (known as the "$\mathbf{P} \overset{?}{=} \mathbf{NP}$ problem"). The Clay Mathematics Institute included it on its list of seven "Millennium Prize Problems" in 2000 and promises a prize of US$1,000,000 for a correct solution.

Exercises.

2.4.2. Exercise. Which of the problems from Exercise 2.2.2 are in the class **NP**? Which of the decision problems formulated in Exercise 2.3.7 belong to **NP**?

Further Exercises and Comments.

2.4.3. The notation **NP** is an abbreviation for "non-deterministically polynomial". The justification for this terminology is that such a problem can be solved efficiently if we first "guess" the correct witness. However, this step is not deterministic (and not possible in practice; the notion of "non-deterministic algorithms" is a purely theoretical concept).

2.4.4. A fascinating aspect of the class **NP** is that it contains some problems that are "at least as difficult" as all other problems in the class. More precisely, there are problems – referred to as "**NP**-complete" – whose efficient solution would also result in an efficient algorithm for every other problem in **NP**. Hundreds, if not thousands, of such problems are known; the above-mentioned three-colorability and the scheduling problem are of this type, as is the popular "minesweeper" game. If $\mathbf{P} \neq \mathbf{NP}$, then none of these problems can be solved efficiently. On the other hand, the existence of an efficient algorithm for a single **NP**-complete problem immediately implies that $\mathbf{P} = \mathbf{NP}$.

2.4.5. $\mathbf{P} \overset{?}{=} \mathbf{NP}$ is not the only open question from complexity theory; there are many other conjectures regarding the relationship between different complexity classes. For example, it is not known whether problem (c) from the examples above (regarding the number of three-colorings) really does not belong to **NP**.

The "complexity zoo" website, founded by the computer scientist Scott Aaronson, contains a list of many common (and not so common) complexity classes. It can currently be found at

`https://complexityzoo.uwaterloo.ca/Complexity_Zoo`.

2.5. Randomized algorithms

Until now we required an algorithm to be **deterministic**, which means that, when started with the same input, it will always follow the same steps and always give the same result.

However, in practice it is often sensible to weaken this condition and to allow the use of **random choices** about how to proceed. Such procedures are called **randomized algorithms**. Depending on the form of the algorithm, a random choice can have different effects:

(a) the output of the algorithm is only correct with a certain probability (**Monte Carlo method**) or

(b) the algorithm always gives a correct answer, but its *running time* can vary (**Las Vegas method**).

Monte Carlo algorithms. Let P be a polynomial in the n variables $X_1, \dots, X_n$, with integer coefficients. This means that there are $C_1, ..., C_k \in \mathbb{Z}$ and $d_{1,1}, ..., d_{n,k} \in \mathbb{N}_0$ such that

$$P = \sum_{i=1}^{k} C_i \cdot X_1^{d_{1,i}} \cdot X_2^{d_{2,i}} \cdots X_n^{d_{n,i}}. \tag{2.5.1}$$

(For example, $P = X^2 + Y^2 - Z^2$ is a polynomial in the three variables X, Y, and Z. Polynomials in *one* variable are studied in much more detail in Section 3.4.)

We can substitute given integers $x_1, \dots, x_n \in \mathbb{Z}$ into such a polynomial P and denote the result by $P(x_1, \dots, x_n)$. (For instance, if $P = X^2 + Y^2 - Z^2$, then $P(3, 4, 5) = 0$.) Let us suppose that, *for* *given values of $x_1, \ldots, x_n$, we can compute the value $P(x_1, \ldots, x_n)$, but we do not know the coefficients of P themselves.* This may seem rather far-fetched, but there actually are many situations where such a polynomial (for example as the *determinant* of a so-called symbolic matrix) is implicitly given, but an actual computation of the coefficients is intractable. (A similar problem will motivate the development of an efficient deterministic primality test in the second part of the book.)

We are now interested in the following question: are there values of $x_1, \ldots, x_n$ such that $P(x_1, \ldots, x_n) \neq 0$? In other words, does P have any non-zero coefficients?

We can try to substitute many different numbers $x_1, \ldots, x_n$ into P; if we find a non-zero value, we have solved the problem. But non-constant polynomials in several variables can have infinitely many zeroes – for example this is the case for $X^2 + Y^2 - Z^2$. So it could be that we just keep picking zeros of P, even though the polynomial is non-zero itself. However, that would be really unlucky, since the next lemma shows that the number of zeros is very small compared to the number of values where P is non-zero.

2.5.2. Lemma (Number of zeros)**.**
Let P be as in (2.5.1)*, let d be the largest exponent $d_{j,i}$ occurring in P, and let $M > 0$. If there is some point at which P is non-zero, then there are at most $n \cdot d \cdot M^{n-1}$ zeros $(x_1, x_2, \ldots, x_n)$ of P with the property that $x_j \in \{1, \ldots, M\}$ for all $j \in \{1, \ldots, n\}$.*

Proof. See Exercise 2.5.5. ■

Note that there are, in total, M^n possibilities to pick integers $x_1, \ldots, x_n$ between 1 and M. So if $M > n \cdot d$, then there is at least one choice that does not lead to a zero of P. If we are looking for a deterministic algorithm, that would not be much help: we would have to try up to $(n \cdot d)^n$ different values to know with certainty that our polynomial is not constant.

But if we choose the values $x_1, \ldots, x_n$ *at random*, then we have a high probability of obtaining a non-zero value, no matter which polynomial we are looking at. For example, set $M := 2 \cdot n \cdot d$. Then the lemma states that the probability of picking a zero $(x_1, x_2, \ldots, x_n)$ (when choosing $x_1, \ldots, x_n$ at random from 1 to M) is bounded from above by $\frac{n \cdot d \cdot M^{n-1}}{M^n} = \frac{n \cdot d}{M} = \frac{1}{2}$.

This leads to the following randomized algorithm:

```
ALGORITHM POLY-ZERO
Input:  A polynomial $P$ as in Lemma 2.5.2.
  1. Choose $x_1, \ldots, x_n$ randomly between 1 and $M$.
  2. If $P(x_1, \ldots, x_n) \neq 0$, then answer “yes”.
  3. Otherwise answer “probably not”.
```

This algorithm has the following properties:

(a) The answer “yes” is always right: i.e. if the algorithm answers “yes”, then the input is a positive instance.

(b) For every positive instance, the probability of giving the (correct) answer “yes” is at least p.

(Here $p = \frac{1}{2}$ for the algorithm POLY-ZERO.)

2.5.3. Definition (Monte Carlo algorithm)**.**
If E is a decision problem, then an algorithm A is called a **Monte Carlo algorithm** for E if there exists a probability $p > 0$ such that A satisfies (a) and (b).

If we apply such an algorithm to an instance I and the output is “yes”, then we know that the instance is positive. Otherwise, we do not know with certainty whether I is a negative instance, but by repeated application of the algorithm to the instance I, we can reduce the error probability very quickly (see Exercise 2.5.6).

From a practical point of view this means that the existence of an efficient Monte Carlo algorithm for the solution of a problem is just as good as the existence of a polynomial-time deterministic algorithm. The class of all decision problems for which there is an efficient Monte Carlo algorithm (i.e. one whose running time is polynomial) is denoted by **RP**.

Las Vegas algorithms. As a further illustration of the advantages of randomization, we study the popular sorting method **Quicksort**. We are given a list of k natural numbers and our goal is to order these numbers by size. For simplicity, let us assume that no number appears several times in the list (although this is not necessary for our methods).

A plausible approach is to begin by finding the smallest number in the list and placing it in the first position, then to search the remaining numbers for the next smallest one, and so on. What is the running time of this method? We have to compare k elements in the first step, only $k-1$ in the second, and so on. So overall the algorithm makes

$$k + (k-1) + \cdots + 2 + 1 = \frac{k(k+1)}{2} = O(k^2)$$

comparisons (here we used Gauss summation; see Exercise 1.1.12(b)). Hence it is efficient. On the other hand, for very large lists, carrying out $O(k^2)$ operations can take a while (imagine ordering a complete dictionary or telephone directory alphabetically).

The Quicksort method allows us to reduce the running time further. Once more, it is based on the "divide and conquer" idea: we split the list in two sublists in such a way that every element from the first list is smaller than every element from the second list. Then we recursively sort these two lists and obtain, in the end, a completely sorted list.

```
ALGORITHM QUICKSORT
Input:  A list $L$.
  1. If $L$ has only one element, then $L$ is already sorted
     and we are done.
  2. Otherwise, choose some element $x$ from the list.
  3. Divide the remaining elements into two lists $L_1$
     and $L_2$ such that $L_1$ contains all elements that are
     smaller than $x$ and $L_2$ contains all larger ones.
  4. Sort $L_1$ and $L_2$ using QUICKSORT. Then output
     the (sorted) list consisting of $L_1$, followed first by $x$
     and then by $L_2$.
```

The running time of this algorithm clearly depends on our choice of the comparison element x. In the *worst* case, x itself is the smallest element of the list L, and in the next step we have to sort a list of size $k-1$, and so on. This means that, in the worst scenario, the running time of QUICKSORT is just as good (or bad) as that of the naive algorithm that we discussed above, namely $O(k^2)$. In the *best* case, however, the two lists L_1 and L_2 have the same length in every step. Then the algorithm requires only $\log k$ levels of recursion to reduce each list to a single element. In every such level, each element of the list is considered at most once, so we have a running time of $O(k \cdot \log(k))$, a clear improvement.

As in the example of zeros of a polynomial there is no simple deterministic method to find the element x such that L_1 and L_2 have the same length. But if we choose x *at random*, then we are likely to find an element that is somewhere in the middle of the list. We refrain from giving the precise calculation here (instead we refer to the further reading material at the end of the chapter), but it turns out that *if we pick the element x at random at every step, then on average QUICKSORT requires only $O(k \cdot \log k)$ total comparisons when sorting a list of length k.* Note that, even in the case of an unfortunate random choice, we always obtain a correctly sorted list; the procedure might just take longer to complete.

Randomized algorithms of this type, where, in contrast to Monte Carlo methods, the answer is *always* correct and where the randomization only affects the running time, are called **Las Vegas algorithms**.

2.5.4. Definition (Las Vegas algorithm)**.**
Suppose that D is a decision problem and that A is a randomized algorithm. Then we call A an **efficient Las Vegas algorithm** for E if

(a) for every input, A gives a correct result as output and

(b) the **average running time** of the algorithm is polynomial in the size of the input.

(Here by average running time we mean the average of the running time function $s(I)$ over all possible executions of the algorithm on the instance I.) The class of all decision problems for which such an algorithm exists is called **ZPP** (**zero-error probability polynomial time**).

From Las Vegas to Monte Carlo and back. The two concepts we have just encountered are closely related. Indeed, it is not difficult to modify any efficient Las Vegas algorithm to obtain an efficient Monte Carlo algorithm A'. To do so, let us denote the expected running time of A on an instance I by $e(I)$. The algorithm A' then runs through the first $2 \cdot e(I)$ steps of A. If the algorithm finishes its execution in this time, we output the (necessarily correct) result. Otherwise we answer "probably not".

It is a consequence of the **Markov inequality** from probability theory (Exercise 2.5.10) that there is a positive probability that A finishes within the first $2 \cdot e(I)$ steps.

With this result it follows that A' is an efficient Monte Carlo algorithm for D because $e(I)$ grows at most polynomially with the length of I by condition (b) in the definition of a Las Vegas algorithm.

Now let us suppose that both the problem D and its dual problem have efficient Monte Carlo algorithms A'_1 and A'_2. Then we can combine these to obtain an efficient Las Vegas algorithm A as follows:

```
                ALGORITHM $A$
Input:  An instance $I$ of the decision problem $D$.
  1. Run $A'_1$ with input $I$.
  2. If $A'_1$ identifies $I$ as a positive instance in Step 1,
     then answer "yes" and we are done.
  3. Otherwise run $A'_2$ with input $I$.
  4. If $A'_2$ identifies $I$ as a negative instance in Step 3,
     then answer "no". We are finished.
  5. Otherwise return to Step 1.
```

Witnesses and Monte Carlo algorithms. Let D be a decision problem in the class **NP**. Recall that for every positive instance of

D there is a *witness*. To solve the problem, it is hence sufficient to find such a witness or to exclude the possibility that such a witness exists. This means that identifying a suitable set of witnesses can be a first step towards developing a Monte Carlo algorithm. To see why, suppose that, for every instance I, the probability of finding a witness for I when picking p at random is larger than some fixed bound. Then we have found a Monte Carlo algorithm for D: we only need to pick a possible witness p at random and check whether it is a witness for I. If I is a negative instance, then this algorithm will always give the correct answer; otherwise there is a positive probability of finding a correct witness and again obtaining the correct answer. We have already seen an example of this idea in the remark after Lemma 2.5.2.

We can try to find a randomized algorithm for the problem COMPOSITES, which we know to belong to **NP**, in this manner. A non-trivial divisor m of n is a witness for the compositeness of n. So our algorithm would pick a number k between 2 and $n-1$ at random and check (using the division algorithm) whether k divides n or not. Unfortunately it is extremely unlikely that we will find a factor of n this way, e.g. when n is the product of two large primes. Indeed, in this case n has only two non-trivial divisors, and the probability of finding one of them is $2/(n-2)$, which is not bounded by a positive constant from below. (We could improve the method a little bit by checking only whether k and n are *coprime*. But this also does not lead to a Monte Carlo algorithm; see Exercise 2.5.8.)

So randomization does not help us here. However, in our study of number theory we shall find other, less obvious witnesses for the compositeness of a natural number. In Section 4.5 we will indeed be able to use these to develop a Monte Carlo algorithm for COMPOSITES in the manner described above. In the second part of the book, this idea is taken one step further: we will describe a witness for the compositeness of a natural number n and show that the *smallest* witness grows at most polynomially in $\log n$. So we can find this witness even deterministically and thus obtain an efficient deterministic algorithm for COMPOSITES (and hence also for PRIMES). But before we are ready for this, we require some further background in number theory, which will be provided in the next two chapters.

Exercises.

2.5.5. Exercise (!). Prove Lemma 2.5.2. (*Hint*: Inductively use the fact that a polynomial of degree d in *one* variable has at most d zeros. See also Corollary 3.4.5.)

2.5.6. Exercise (!). To work out how many times we must run a Monte Carlo algorithm to obtain a small probability of error, it is useful to imagine throwing a coin. Suppose that, for each throw, the probability of throwing "heads" is p and the probability of throwing "tails" is $q = 1 - p$.

(a) What is the probability of throwing "tails" n times in a row?

(b) In the case $q = 1/2$, how many throws do we need to ensure that the probability from part (a) is at most 0.0001%?

(c) We now throw the coin until we get "heads" for the first time. What is the average required number of throws?

Hint: Use the following consequence of Exercise 1.1.12(d): for every real number $x \in \mathbb{R}$ with $|x| < 1$,

$$\sum_{k=1}^{\infty} k \cdot x^{k-1} = \frac{1}{(1-x)^2}.$$

(d) Conclude that on average, for a non-constant polynomial P, the algorithm POLY-ZERO requires at most two executions to find a non-zero value of P.

Further Exercises and Comments.

2.5.7. In our presentation of randomization we ignored an important question: can an algorithm really pick a number at random in practice? I.e. what importance do randomized algorithms really have for the practical solution of problems? Here we remark only that randomization really does get used successfully in everyday applications (e.g. in internet security) and refer the reader to further literature on algorithm theory; see below.

2.5.8. Exercise. Consider the following randomized algorithm:

Given $n \in \mathbb{N}$ with $n \geq 2$, randomly choose a number $m \in \{1, \ldots, n-1\}$. If $\gcd(n, m) \neq 1$, output "n is composite". Otherwise output "n is perhaps prime".

(a) Show: if n is the product of two prime numbers p and q, then there are exactly $(p-1) + (q-1)$ numbers between 1 and $n-1$ that have a common prime factor with n.

(b) Conclude that the above method is not a Monte Carlo algorithm for COMPOSITES.

2.5.9. Exercise (P). Implement the method from Exercise 2.5.8 in a common programming language. Run it for each of the (composite) numbers 120, 143, 7 327 883, and 1 726 374 899 084 624 209 until either a number is found that is not coprime to n or until the algorithm has been completed 1 000 000 times.

2.5.10. Exercise. In this exercise we use the standard terminology of probability theory. Let X be a random variable that takes only positive values. Furthermore let μ be the *expected value* of X.

Let $a > 1$. Prove the **Markov inequality**: the probability that X is larger than $a \cdot \mu$ is at most $1/a$.

2.5.11. Exercise. Let $k \in \mathbb{N}$. Show that the definition of the class **RP** does not change if we replace (b) by the following: for a positive instance of length n, the probability of obtaining the correct answer is at least $1/n^k$.

2.5.12. Exercise. Show that $\mathbf{RP} \subseteq \mathbf{NP}$. (*Hint:* Consider the sequence of random choices that the Monte Carlo algorithm makes during its execution. If the instance is positive, then there is such a sequence for which the eventual output is "yes".)

In particular we have $\mathbf{P} \subseteq \mathbf{ZPP} \subseteq \mathbf{RP} \subseteq \mathbf{NP}$. It is not known whether any of these inclusions are proper.

Further reading

The book *Introduction to Algorithms* [**CLR**] provides a comprehensive but gentle introduction to the theory and practice of algorithms. A more formal development of the foundations of algorithm theory can be found in the books *Elements of the Theory of Computation* [**LPa**] as well as *Introduction to Automata Theory, Languages, and Computation* [**HMU**]. We would also like to recommend the more advanced text *Computational Complexity* by Papadimitriou [**P**]. This excellent book describes a plethora of aspects of complexity theory and contains chapters covering $\mathbf{P} \overset{?}{=} \mathbf{NP}$, randomized algorithms, cryptography, and much more. For example, our treatement of the algorithm POLY-ZERO is inspired by the discussion of symbolic determinants in this book. Introductory books of probability theory, such as [**Ro**], provide relevant mathematical background for studying randomized algorithms.

Chapter 3

Foundations of number theory

In some sense this chapter is the heart of the first part of the book, as it is here that we encounter the most important ingredients of the AKS algorithm. We have already learned about division with remainder in Section 1.2. To gain further familiarity with the concept, we now study **modular arithmetic**, which describes how remainders behave under addition and multiplication. As we shall see, modular arithmetic with respect to a prime has particularly nice properties, which is important for the development of primality tests. An example is **Fermat's Little Theorem**, which we prove (together with two generalizations, the Fermat-Euler Theorem and Lagrange's Theorem) in Section 3.2. This result allows us to develop our first primality test in in Section 3.3.

At the end of the chapter, we turn our attention to polynomials. After introducing common notations and simple rules for calculations, we explain how to perform modular arithmetic with polynomials. This concept plays a crucial role in the second part of the book.

3.1. Modular arithmetic

When carrying out **modular arithmetic** with respect to a natural number $n \geq 2$, we treat numbers that leave the same remainder after division through n as if they were equal. As an example, consider calculations that involve the time of day. If it is 10:00 in the morning and we would like to know what time it will be in seventy-nine hours, we add 10 and 79 and compute the remainder after division by 24, which is 17. So the answer is "17:00" (or "5 pm").

To illustrate the concept further, consider the partition of $\mathbb{Z}$ into **odd** and **even** numbers. (Recall that the former are divisible by 2, while the latter leave remainder 1 when divided by 2.)

The sum of two even numbers is again even, as is the sum of two odd numbers; on the other hand the sum of an even and an odd number is always odd. The product of an even number with any other number is even; the product of two odd numbers is odd (Exercise 1.1.10).

If we would like to know whether the result of some calculation is divisible by two or not, this means that we only have to remember in each step which of the numbers are even, resp. odd. For example, we can see that $2^{23} \cdot 5 - 27^5 + 33^2 - 1$ is odd without having to compute this (rather large) number explicitly. (The first number is even and the remaining three numbers are odd, so we are left with the sum of one even and three odd numbers, which is always odd.)

We now formulate this idea in general.

3.1.1. Definition (Congruence)**.**
Let $n \geq 2$ be a natural number. If a and b are integers that have the same remainder when divided by n, we write $a \equiv b \pmod{n}$. In this case we say: a **is congruent to** b **modulo** n.

Example. Two non-negative integers are congruent modulo 10 if and only if they have the same final digit; for example $13 \equiv 33 \pmod{10}$. We also have $-2 \equiv 38 \pmod{10}$, since both numbers leave remainder 8 when they are divided by 10.

The notion of congruence behaves well under elementary arithmetic. If, for example, we replace one of the terms in a sum by a congruent number, then the new result will be congruent to the old one. More precisely:

3.1.2. Lemma (Rules of modular arithmetic)**.**
Let $n \geq 2$ and let a, b, c, and d be arbitrary integers.

(a) *If $a \equiv b$ and $b \equiv c \pmod{n}$, then also $a \equiv c \pmod{n}$.*

(b) *The numbers a and b are congruent modulo n if and only if n divides the difference $a - b$.*

(c) *If $a \equiv b$ and $c \equiv d \pmod{n}$, then $a + c \equiv b + d$, $a - c \equiv b - d$, and $a \cdot c \equiv b \cdot d \pmod{n}$.*

(d) *If $a \equiv b \pmod{n}$, then $a^k \equiv b^k \pmod{n}$ for all $k \in \mathbb{N}$.*

Proof. The first claim follows directly from the definition.

For (b), let $r_a, r_b \in \mathbb{N}_0$ denote the remainders of a, resp. b, after division by n. So there are $s, t \in \mathbb{Z}$ with $a = s \cdot n + r_a$ and $b = t \cdot n + r_b$, and by definition r_a and r_b are smaller than n.

If $a \equiv b \pmod{n}$, which means that $r_a = r_b$, we see that

$$a - b = s \cdot n - t \cdot n = (s - t) \cdot n,$$

and $a - b$ is divisible by n as claimed. If, conversely, $a - b$ is divisible by n, then $r_a - r_b = a - b + (t - s) \cdot n$ is also divisible by n. But $-n < r_a - r_b < n$, so it follows that $r_a - r_b = 0$. Hence the remainders r_a and r_b are equal, as desired.

Now we can use part (b) to prove the remaining claims in the lemma. Indeed, if $a \equiv b$ and $c \equiv d \pmod{n}$, then we know that $a - b$ and $c - d$ are divisible by n. Thus $(a - b) + (c - d) = (a + c) - (b + d)$ is also divisible by n, and therefore $a + c \equiv b + d \pmod{n}$, as claimed. In the same manner we prove the statement about differences and products, and readers are invited to prove (d) using mathematical induction (a nice exercise). ■

Example. Lemma 3.1.2 tells us that we can add, subtract, multiply, and take powers modulo n just as if we were working with regular

$\oplus$	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

$\odot$	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

Figure 3.1. Addition and multiplication modulo 10.

integers. As an example, we calculate modulo 2:

$$2^{23} \cdot 5 - 27^5 + 33^2 - 1 \equiv 0^{23} \cdot 1 - 1^5 + 1^2 - 1 = 0 - 1 + 1 - 1 = -1 \equiv 1.$$

Exercises 3.1.19 and 3.1.20 illustrate that modular arithmetic can even be used to answer questions that, at first glance, have no connection to “division with remainder” at all.

Every integer $a \in \mathbb{Z}$ is congruent modulo n to exactly one of the numbers $0, 1, \ldots, n-1$, namely to its remainder after division by n. When doing modular arithmetic, this means that the result of every operation can be replaced by its remainder. For this reason, we sometimes use the notation $M \bmod n$ for the remainder of M after division by n. For example, $(5+7) \bmod 8 = 4$.

We can also think of modular arithmetic as defining a “new” addition $\oplus$ and a “new” multiplication $\odot$ on the set $\{0, \ldots, n-1\}$ by $a \oplus b := (a+b) \bmod n$ and $a \odot b := a \cdot b \bmod n$. For these operations we can write down complete addition and multiplication tables, as is shown in Figure 3.1 for addition and multiplication modulo 10. (See also Comment 3.1.25.)

So addition, subtraction, and multiplication modulo n work just according to our intuition. We now look at division, where things turn out to be rather different. Recall the basic facts about division in the familiar number systems: in $\mathbb{Q}$ and $\mathbb{R}$ we can divide by any number except zero, but 1 and -1 are the only integers by which we can always divide in $\mathbb{Z}$. (Division by other integers will usually not yield an integer again.) We now ask the same question about modular arithmetic: what conditions on $n \geq 2$ and $a \in \mathbb{Z}$ are needed so that it is possible to divide every number $x \in \mathbb{Z}$ by a modulo n? That is, when does the congruence

$$y \cdot a \equiv x \pmod{n} \tag{3.1.3}$$

have a solution $y \in \mathbb{Z}$ for every $x \in \mathbb{Z}$? Or, to rephrase the question in yet another way, when does the a-th column of the multiplication table contain all possible remainders $0, 1, \ldots, n-1$? If we take a look at the table for $n = 10$ (Figure 3.1), we see that this is the case for the numbers 1, 3, 7, and 9, but not for 0, 2, 4, 5, 6, and 8. This suggests that a should be coprime to n if we wish to be able to carry out arbitrary division by a modulo n.

3.1.4. Definition and Theorem (Division modulo n)**.**
Let $n \geq 2$ and $a \in \mathbb{Z}$. If a and n are coprime, then for every integer x there is an integer y such that

$$y \cdot a \equiv x \pmod{n}. \tag{3.1.5}$$

Furthermore, for every $a \in \mathbb{Z}$, the congruence $y \cdot a \equiv 1 \pmod{n}$ has a solution y if and only if a and n are coprime. In this case we say that y is a (multiplicative) ***inverse*** *of a modulo n.*

Proof. First suppose that a and n are coprime. We must show that there are integers y and m such that $y \cdot a + m \cdot n = x$. This reminds us of Bézout's Lemma (Theorem 1.4.2), which tells us that there are numbers $s, t \in \mathbb{Z}$ such that $s \cdot a + t \cdot n = \gcd(a, n) = 1$. We set $y := s \cdot x$ and $m := t \cdot x$, and then we have proved the first claim of the theorem. The second part is nothing but a restatement of Corollary 1.4.3. ■

The proof of Theorem 3.1.4 shows us not only that the desired number y exists, but also how we can explicitly compute it. Indeed, we saw that the representation of $\gcd(a, n) = 1$ in terms of a and n from Bézout's Lemma can be calculated using the Euclidean algorithm. As an example, we calculate the inverse of 5 modulo 73. First apply the

Euclidean algorithm:

$$73 = 14 \cdot 5 + 3; \quad 5 = 1 \cdot 3 + 2; \quad 3 = 1 \cdot 2 + 1.$$

Then we substitute backwards as in Section 1.4:

$$\begin{aligned} 1 &= 3 - 1 \cdot 2 = 3 - 1 \cdot (5 - 1 \cdot 3) \\ &= 2 \cdot 3 - 1 \cdot 5 = 2 \cdot (73 - 14 \cdot 5) - 1 \cdot 5 \\ &= 2 \cdot 73 - 29 \cdot 5 \equiv 44 \cdot 5 \pmod{73}. \end{aligned}$$

So we see that 44 is an inverse of 5 modulo 73.

3.1.6. Corollary (Cancellation rule)**.**
Let $n \geq 2$ and let a, b, and c be integers such that $a \cdot b \equiv a \cdot c \pmod{n}$. If a and n are coprime, then $b \equiv c \pmod{n}$. In particular, the inverse of a is "unique modulo n"; i.e. all inverses have the same remainder when divided by n.

Proof. The idea is simple: since a and n are coprime, we can divide by a on both sides of the given congruence. More formally, we know by Theorem 3.1.4 that there is an inverse y of a modulo n. Then "dividing by a modulo n" means multiplying by the inverse y. Doing this on both sides of the congruence, we conclude that

$$y \cdot a \cdot b \equiv y \cdot a \cdot c \pmod{n}.$$

Since $y \cdot a \equiv 1 \pmod{n}$, it follows that

$$b = 1 \cdot b \equiv y \cdot a \cdot b \equiv y \cdot a \cdot c \equiv 1 \cdot c = c \pmod{n},$$

and we have proved the first claim.

If y and y' are both inverses of a modulo n, then $y \cdot a \equiv 1 \equiv y' \cdot a$. Using the cancellation rule we have just proved, it follows that $y \equiv y'$ as claimed. ■

Examples. We have $45 \equiv 15 \pmod{6}$ because $45 - 15 = 30$ is divisible by 6. Since 5 and 6 are coprime, we are allowed to cancel 5 from both sides and get $9 \equiv 3 \pmod{6}$ (which is true). But if we divide by 3 (not coprime to 6) on both sides, we get the numbers 15 and 5, and these are *not* congruent modulo 6. Hence the requirement that a and n are coprime is really essential in Corollary 3.1.6.

It follows from Theorem 3.1.4 that modular arithmetic with respect to a prime is particularly nice. Indeed, a number a is coprime to p if and only if it is not divisible by p (Exercise 1.2.12), i.e. if $a \not\equiv 0 \pmod{p}$. This means that, modulo p, we can divide by every number that is not congruent to zero – essentially just as in the rational and real numbers. As a result, many properties of these number systems also hold for modular arithmetic with respect to a prime; we shall learn about some of these throughout the chapter.

Calculations modulo a composite number n, on the other hand, are far more uncomfortable: we can only divide and cancel by integers that are coprime to n, and sums and differences of such numbers will not again be coprime to n in general. (See Exercise 3.1.13.) In the following, we will again and again come across important differences between primes and composite numbers when it comes to modular arithmetic. These differences will help us to devise methods of distinguishing between these two types of natural numbers.

To conclude the section, we present the famous **Chinese Remainder Theorem**. It states that calculations modulo a composite number $n = n_1 \cdot n_2$ can be reduced to calculations modulo n_1 and modulo n_2 provided that the two numbers are coprime. Essentially this tells us that we can understand modular arithmetic with respect to a composite number n quite well *provided that we know how to factorize* n. (This theorem will only be used in Section 4.5 and in some exercises; it is not required for the AKS algorithm itself.)

3.1.7. Theorem (Chinese Remainder Theorem).
Let $n_1, n_2 \geq 2$ be coprime natural numbers and set $n := n_1 \cdot n_2$.

Then two integers are congruent modulo n if and only if they are congruent modulo n_1 and congruent modulo n_2. Conversely, for any two integers a_1 and a_2, there exists some $x \in \mathbb{Z}$ simultaneously satisfying the two congruences $x \equiv a_1 \pmod{n_1}$ and $x \equiv a_2 \pmod{n_2}$.

Remark. Often the theorem is formulated more generally for products of arbitrarily many pairwise coprime numbers. This version follows from ours by induction; see Exercise 3.1.21.

Proof of the Chinese Remainder Theorem. Let $b, c \in \mathbb{Z}$. If b is congruent to c modulo n, then it follows from the basic properties of division in $\mathbb{Z}$ that also $b \equiv c \pmod{n_1}$ and $b \equiv c \pmod{n_2}$ (for details, see Exercise 3.1.10).

If conversely $b \equiv c \pmod{n_1}$ and $b \equiv c \pmod{n_2}$, then we deduce from Lemma 3.1.2 that $b - c$ is divisible by n_1 and n_2. This means that $b - c$ is a common multiple of b and c and hence it is a multiple of $\operatorname{lcm}(n_1, n_2)$. Now by hpothesis n_1 and n_2 are coprime, so $\operatorname{lcm}(n_1, n_2) = n_1 \cdot n_2 = n$ (Exercise 1.3.9). Hence we have $b \equiv c \pmod{n}$, and the proof of the first claim is complete.

We could deduce the second part of the theorem from the first, just by counting (Exercise 3.1.18). Instead we give a direct proof, which also shows us how to find x. Since n_1 and n_2 are coprime, Bézout's Lemma tells us that there are numbers m_1 and m_2 such that $m_1 \cdot n_1 + m_2 \cdot n_2 = 1$. Thus we see that $m_1 \cdot n_1 \equiv 1 \pmod{n_2}$ and $m_2 \cdot n_2 \equiv 1 \pmod{n_1}$. If we set $x := a_2 \cdot m_1 \cdot n_1 + a_1 \cdot m_2 \cdot n_2$, then

$$x \equiv a_1 \cdot m_2 \cdot n_2 \equiv a_1 \cdot 1 = a_1 \pmod{n_1}$$

and similarly $x \equiv a_2 \pmod{n_2}$, as claimed. ■

Exercises.

3.1.8. Exercise. Work out complete addition and multiplication tables modulo 3 and modulo 7.

3.1.9. Exercise. What is the remainder of 9! when dividing by 10, of 10! when dividing by 11, of 11! when dividing by 12, of 12! when dividing by 13? For a general composite number n, what number is $(n-1)!$ congruent to modulo n?

3.1.10. Exercise (!). Let a and b be integers and let $m, n \geq 2$ be natural numbers such that m divides n. Prove the following: if $a \equiv b \pmod{n}$, then also $a \equiv b \pmod{m}$. Does the converse hold?

3.1.11. Exercise. Let $n \geq 2$. When is $\sum\limits_{i=0}^{n-1} i \equiv 0 \pmod{n}$? When is $\sum\limits_{i=0}^{n-1} i^2 \equiv 0 \pmod{n}$? (See Exercise 1.1.12.)

3.1.12. Exercise. Let $n \geq 2$ and let a and x be arbitrary integers. Once more we study the congruence $a \cdot y \equiv x \pmod{n}$; i.e. we ask under what conditions x is divisible by a modulo n. By Theorem 3.1.4 this is always possible if a and n are coprime. Develop a condition on x, n, and a under which the congruence has solutions even if $\gcd(a, n) \neq 1$. In particular, show that the solution y is *unique modulo* n if and only if a and n are coprime.

(*Hint:* Set $d := \gcd(a, n)$ and distinguish two cases, namely whether x is a multiple of d or not.)

3.1.13. Exercise (!). Let $n \geq 2$. Two numbers $a, b \in \mathbb{Z}$ are called **zero divisors modulo** n if $a \cdot b \equiv 0 \pmod{n}$, but neither a nor b is congruent to 0 modulo n. For example, 2 is a zero divisor modulo 4 because $2 \cdot 2 = 4 \equiv 0$. Find pairs of zero divisors modulo 6 and modulo 10. Show that there are no zero divisors modulo 3 or modulo 5. Then prove the following theorem: *there are zero divisors modulo n if and only if n is composite.*

3.1.14. Exercise (!). Let $a, k, n \in \mathbb{N}_0$ with $n \geq 2$.

(a) Show that there is an efficient algorithm to compute the remainder of a^k when dividing by n. Recall that "efficiency" means that the number of elementary instructions should grow at most polynomially with $\log n$, $\log a$, and $\log k$. (*Hint:* Use the power algorithm from Section 2.3, but in each step carry out a division with remainder by n.)

(b) Let a and n be coprime. Show that the method of computing the inverse of a modulo n, as described after Theorem 3.1.4, is efficient.

3.1.15. Exercise (P). Implement the algorithms from Exercise 3.1.14 in a common programming language. Use these to compute a^k and the inverse of a for $n = 1\,726\,374\,887$, $a = 3$, and $k = 1\,726\,374\,885$. (Many pocket calculators support modular arithmetic. There are also many online calculators with such functions on the World Wide Web; for example the "Big Number Calculator" at

`http://world.std.com/~reinhold/BigNumCalc.html`

performs modular arithmetic for numbers with arbitrarily many digits.)

3.1.16. Exercise. Find $x \in \mathbb{Z}$ with $x \equiv 1 \pmod{5}$ and $x \equiv 3 \pmod{13}$. Can x be chosen in such a way that also $x \equiv 2 \pmod{7}$?

3.1.17. Exercise. Let $n_1, n_2 \geq 2$ and $a_1, a_2 \in \mathbb{Z}$. The Chinese Remainder Theorem tells us that the congruences

$$x \equiv a_1 \pmod{n_1} \quad \text{and} \quad x \equiv a_2 \pmod{n_2}$$

have a common solution if n_1 and n_2 are coprime to each other. When are there solutions even if n_1 and n_2 are not coprime?

3.1.18. Exercise. Let $n = n_1 \cdot n_2$ be as in the Chinese Remainder Theorem. Then there are exactly n different pairs (r_1, r_2) of integers such that $0 \leq r_1 < n_1$ and $0 \leq r_2 < n_2$.

Use this observation to deduce the second part of the Chinese Remainder Theorem from the first. (*Hint:* For a number x, let $r_1(x)$ be the remainder of x when dividing by n_1, and let $r_2(x)$ be the remainder when dividing by n_2. If x and x' are different modulo n, then the pairs $(r_1(x), r_2(x))$ and $(r_1(x'), r_2(x'))$ are also different from each other.)

Further Exercises and Comments.

3.1.19. Exercise. Show, by working modulo 5, that the equation

$$x^2 = 5y - 2$$

does not have integer solutions. (*Hint:* Consider the possible remainders of a perfect square modulo 5.)

3.1.20. Exercise. Consider the equation

$$x^2 + 2 = y^3,$$

where $x, y \in \mathbb{Z}$. By studying this equation modulo 4, show that every solution must satisfy $x^2 \equiv 1$ and $y^3 \equiv -1 \pmod{4}$. (In particular, x and y must both be odd.)

Such arguments, and much more subtle considerations, are often encountered in the theory of **Diophantine equations**, a beautiful area of number theory. For more information see, for example, [**HW**, Section 8].

3.1.21. Exercise. Let $t \geq 2$. State and prove a version of the Chinese Remainder Theorem for numbers $n = n_1 \cdot n_2 \cdots n_t$, provided that these factors are pairwise coprime.

3.1.22. A **complete set of residues (CSR) modulo** n is a set $R \subseteq \mathbb{Z}$ with the property that every integer is congruent to **exactly one** element of R modulo n. For example, $\{0, 1, 2, \ldots, n-1\}$ is a CSR modulo n, but there are infinitely many others. For example, $\{3, 6, 9, 12, 15\}$ is a CSR modulo 5. Every CSR modulo n contains exactly n elements.

A **reduced set of residues (RSR) modulo** n is a set T of integers coprime to n such that every integer that is coprime to n is congruent to exactly one element of T. For example, the set $\mathrm{cp}(n)$ of numbers between 1 and $n-1$ that are coprime to n is an RSR modulo n, and $\{7, 35\}$ is an

RSR modulo 6. Any two different RSRs modulo n must have the same number of elements; in the next section we will see how many exactly.

3.1.23. Exercise. Let $n \geq 2$ and $a \in \mathbb{Z}$. Also let $R = \{r_1, r_2, \ldots, r_n\}$ be a CSR modulo n; we study the set $aR = \{a \cdot r_1, a \cdot r_2, \ldots, a \cdot r_n\}$.

Prove the following: aR is a CSR modulo n if and only if a and n are coprime. Prove that the same claim also holds for reduced sets of residues.

3.1.24. We have seen that the rational and real numbers have many properties in common with the integers modulo p. Mathematically it thus makes sense to combine these concepts in one definition.

Let K be a set on which an addition and a multiplication are defined (in particular we assume that sums and products of elements in K again belong to K). Suppose that K contains a **zero**, i.e. an element $0 \in K$ such that $a + 0 = a$ for all $a \in K$. Similarly, suppose that there is a **one**, i.e. an element $1 \in K$ with $1 \neq 0$ such that $a \cdot 1 = a$ for all $a \in K$. (In particular, K contains at least two elements.)

We also assume that $K \backslash \{0\}$ is closed under multiplication, that addition and multiplication satisfy the usual associative, commutative, and distributive laws, and that we can subtract any element of K from any other, as well as divide by any element except zero. (This is slightly informal and we leave it to the reader to think about what these statements should mean precisely.)

A set K with these properties is called a **field**. Theorems about real numbers whose proofs only use the properties stated above are then automatically true for every field, so that we do not need to prove them separately each time.

For example, show that:

(a) If K is a field, then the elements 0 and 1 are uniquely determined.

(b) If K is a field, then $a \cdot 0 = 0$ for all $a \in K$.

(c) A field contains no **zero divisors**, i.e. if $x, y \in K$ are such that $x \cdot y = 0$, then $x = 0$ or $y = 0$.

3.1.25. At the beginning of the section we said that, in modular arithmetic, we consider numbers that are congruent to be "the same" in some sense. Working with **congruence classes** formalizes this idea in an elegant way. Let $n \geq 2$ and $a \in \mathbb{Z}$. The set $\overline{a}$ of all integers that are congruent to a modulo n is called the **congruence class** of a (modulo n):

$$\overline{a} := \{b \in \mathbb{Z} : a \equiv b \pmod{n}\} = \{a + m \cdot n : m \in \mathbb{Z}\}.$$

We also say that a is a **representative** of the congruence class $\overline{a}$.

Now we can define an addition and multiplication on the set of all congruence classes modulo n by $\overline{a} + \overline{b} := \overline{a+b}$ and $\overline{a} \cdot \overline{b} := \overline{a \cdot b}$. From Lemma 3.1.2, we know that $\overline{a}+\overline{b}$ and $\overline{a}\cdot\overline{b}$ really depend only on the classes $\overline{a}$ and $\overline{b}$ and *not* on the choice of representatives a and b.

In other words, we combine all integers congruent to a to a new "number" (namely the congruence class $\overline{a}$). We think of the set of congruence classes as a new number system, with very natural notions of addition and multiplication. This is the conceptually cleanest way to define modular arithmetic. However, we will not use this somewhat abstract point of view in the main text of the book.

3.2. Fermat's Little Theorem

Having considered addition, subtraction, multiplication, and division modulo a number n, we now turn our attention to taking powers. For example, the powers of 3 modulo 7 are

$$3^0 = 1, \quad 3^1 = 3, \quad 3^2 = 9 \equiv 2, \quad 3^3 = 27 \equiv 6, \quad 3^4 = (3^2)^2 \equiv 2^2 = 4,$$
$$3^5 \equiv 5, \quad 3^6 \equiv 1, \quad 3^7 \equiv 3, \quad 3^8 \equiv 6, \quad 3^9 \equiv 4, \quad \ldots.$$

Similarly, we compute the powers of 3 modulo 8 to be

$$1, \quad 3, \quad 1, \quad 3, \quad 1, \quad \ldots \quad \pmod 8.$$

In both examples we arrive at 1 at some point, and the sequence repeats from then on. This is true more generally and leads us to an important concept: the **order** of a number modulo n.

3.2.1. Definition and Lemma (Order modulo n)**.**
*Let $n \geq 2$ and let $a \in \mathbb{Z}$ be coprime to n. Then there is a natural number k such that $a^k \equiv 1 \pmod n$. The smallest such number is called the **order of** a **modulo** n and is denoted by* $\operatorname{ord}_n(a)$.

For integers $k_1, k_2 \geq 0$, we have that $a^{k_1} \equiv a^{k_2} \pmod n$ if and only if k_1 and k_2 differ by a multiple of $\operatorname{ord}_n(a)$.

Proof. There are only finitely many possible remainders when dividing by n, namely the numbers $0, 1, \ldots, n-1$. Hence there must be two numbers k_1 and k_2 such that $k_2 > k_1$ and $a^{k_1} \equiv a^{k_2} \pmod n$. Then

$$a^{k_1} \equiv a^{k_2} = a^{k_1} \cdot a^{k_2-k_1} \pmod n.$$

n	orders modulo n	n	orders modulo n
2	1	7	1, 3, 6, 3, 6, 2
3	1, 2	8	1, 2, 2, 2
4	1, 2	9	1, 6, 3, 6, 3, 2
5	1, 4, 4, 2	10	1, 4, 4, 2
6	1, 2	11	1, 10, 5, 5, 5, 10, 10, 10, 5, 2

Figure 3.2. The orders of all numbers from 1 to $n-1$ that are coprime to n, for n from 2 to 11.

Since a and n are coprime, we can cancel out a^{k_1} on both sides of the congruence by Corollary 3.1.6. Thus $a^{k_2-k_1} \equiv 1 \pmod{n}$. In particular we have proved the first claim, and there is a smallest natural number $k = \operatorname{ord}_n(a) \geq 1$ with $a^k \equiv 1$.

We now divide $k_2 - k_1$ by k with remainder; that is, we write $k_2 - k_1 = s \cdot k + r$ with $s \in \mathbb{Z}$ and $0 \leq r < k$. Then

$$1 \equiv a^{k_2-k_1} = a^{s\cdot k+r} = \left(a^k\right)^s \cdot a^r \equiv 1^s \cdot a^r = a^r.$$

This forces $r = 0$ because otherwise k could not be the smallest natural number with $a^k \equiv 1$. Hence $k_2 - k_1 = s \cdot k$ and $k_2 - k_1$ is divisible by k as claimed. The converse is left to the reader in Exercise 3.2.13(a). ■

Examples. It follows from our calculations above that $\operatorname{ord}_7(3) = 6$ and $\operatorname{ord}_8(3) = 2$. If we know the order of a number a modulo n, then the remainders of even very large powers of a can be computed very easily using our lemma. For example

$$3^{9001} = 3^{6\cdot1500+1} = (3^6)^{1500} \cdot 3^1 \equiv 1^{1500} \cdot 3 = 3 \pmod{7}.$$

The order of a modulo n can never be larger than $n-1$ since there are at most $n-1$ numbers that are pairwise different modulo n and coprime to n. What else can we say? For each number n from 2 to 11, let us compute the order of every number a from 2 to $n-1$ that is coprime to n (Figure 3.2). We notice that, for the primes $n = 2, 3, 5, 7, 11$, all orders are divisors of $n-1$. The fact that this is true for every prime number is known as **Fermat's Little Theorem**.

$$\begin{array}{c}
1 \\
1 \quad 1 \\
1 \quad \boxed{2} \quad 1 \\
1 \quad \boxed{3} \quad \boxed{3} \quad 1 \\
1 \quad \boxed{4} \quad 6 \quad \boxed{4} \quad 1 \\
1 \quad \boxed{5} \quad \boxed{10} \quad \boxed{10} \quad \boxed{5} \quad 1 \\
1 \quad \boxed{6} \quad 15 \quad 20 \quad 15 \quad \boxed{6} \quad 1 \\
1 \quad \boxed{7} \quad \boxed{21} \quad \boxed{35} \quad \boxed{35} \quad \boxed{21} \quad \boxed{7} \quad 1 \\
1 \quad \boxed{8} \quad 28 \quad \boxed{56} \quad 70 \quad \boxed{56} \quad 28 \quad \boxed{8} \quad 1 \\
1 \quad \boxed{9} \quad \boxed{36} \quad 84 \quad \boxed{126} \quad \boxed{126} \quad 84 \quad \boxed{36} \quad \boxed{9} \quad 1 \\
1 \quad \boxed{10} \quad 45 \quad \boxed{120} \quad \boxed{210} \quad 252 \quad \boxed{210} \quad \boxed{120} \quad 45 \quad \boxed{10} \quad 1 \\
1 \quad \boxed{11} \quad \boxed{55} \quad \boxed{165} \quad \boxed{330} \quad \boxed{462} \quad \boxed{462} \quad \boxed{330} \quad \boxed{165} \quad \boxed{55} \quad \boxed{11} \quad 1
\end{array}$$

Figure 3.3. The first twelve rows of Pascal's triangle. In the $(n+1)$-th row, we have emphasized (for $n \geq 2$) the binomial coefficients that are divisible by n.

3.2.2. Theorem (Fermat's Little Theorem)**.**
Let p be prime and let $a \in \mathbb{Z}$. Then

$$a^p \equiv a \pmod{p}. \tag{3.2.3}$$

In particular, if a is not a multiple of p, then $a^{p-1} \equiv 1 \pmod{p}$. Thus $\mathrm{ord}_p(a)$ is a divisor of $p-1$.

We have already encountered the case $p = 3$ of Theorem 3.2.2 in Example 1.1.5. There, the idea of the proof was to expand $(n+1)^3$ in the induction step, obtaining $n^3 + 3n^2 + 3n + 1$. Modulo 3 this is the same as $n^3 + 1$, and we can apply the induction hypothesis.

So the key fact is that the binomial coefficients that appear in the expansion are divisible by 3. To prove Fermat's Little Theorem, we ask ourselves whether this is true more generally. Taking a look at the first few rows of Pascal's triangle (Figure 3.3), we might suspect that for a prime p every "interior" binomial coefficient

$$\binom{p}{k}, \quad \text{where} \quad 1 \leq k \leq p-1,$$

is divisible by p (while this is not true for composite numbers).

3.2.4. Lemma (Divisors of binomial coefficients)**.**
If p is prime and $1 \leq k \leq p-1$, then

$$\binom{p}{k} \equiv 0 \pmod{p}.$$

Proof. We use the explicit formula (1.1.16) for binomial coefficients using factorials from Exercise 1.1.15. We can rewrite this equation as follows:

$$k!\,(p-k)!\binom{p}{k} = p!\,. \tag{3.2.5}$$

Now p is a divisor of the right-hand side of (3.2.5) and hence also divides the left-hand side. Since p is prime and each of the numbers $1, 2, \ldots, k$ is smaller than p, we know that p does not divide $k!$. (Recall Corollary 1.3.5.) For the same reason, p does not divide $(p-k)!$. So, again by Corollary 1.3.5, it follows that p must divide the final term in the product on the right-hand side and therefore $p \mid \binom{p}{k}$ as claimed. ■

Proof of Fermat's Little Theorem. It suffices to prove the theorem for non-negative numbers a (otherwise we replace a by a non-negative number congruent to a, for example its remainder when dividing by p).

Just as in Example 1.1.5, we now use mathematical induction to prove that $a^p \equiv a \pmod{p}$ for all $a \in \mathbb{N}_0$. For $a = 0$ we know that $a^p = 0 = a$; this is the basis of the induction.

Now let $a \in \mathbb{N}_0$ be such that $a^p \equiv a \pmod{p}$. By the binomial theorem, we have

$$(a+1)^p = \sum_{i=0}^{p} \binom{p}{i} a^{p-i}.$$

Lemma 3.2.4 tells us that the binomial coefficients $\binom{p}{1}, \ldots, \binom{p}{p-1}$ are all divisible by p. So if we reduce the above equation modulo p, then we are only left with the terms for $i = 0$ and $i = p$. By the induction hypothesis, we also see that $a^p \equiv a \pmod{p}$, so overall

$$(a+1)^p \equiv a^p + 1 \equiv a + 1 \pmod{p}.$$

This completes the induction step and proves the first part of the theorem.

If a is not divisible by p, then we can cancel a on both sides of (3.2.3) by Corollary 3.1.6. We obtain that $a^{p-1} \equiv 1 \pmod{p}$ as claimed. By Lemma 3.2.1, it follows that $\operatorname{ord}_p(a)$ divides $p-1$. ■

The theorems of Fermat-Euler and Lagrange. We now discuss two generalizations of Fermat's Little Theorem that are needed to study RSA encryption and certain primality tests later in the book. However, their knowledge will not be required for the AKS algorithm.

Let us return to the table in Figure 3.2. We notice, for example, that for $n = 9$ the orders of the six numbers coprime to n are all divisors of 6. Similarly, for $n = 10$ the orders of the four numbers coprime to n are all divisors of 4. Overall, for every number n in our table, the possible orders are always divisors of *the number of integers from* 1 *to* $n-1$ *that are coprime to* n. We now introduce a name for this value:

3.2.6. Definition (Euler's totient function)**.**
Let $n \geq 2$ and let $\operatorname{cp}(n)$ denote the set of all numbers from 1 to $n-1$ that are coprime to n. We write $\varphi(n)$ for the number of elements of $\operatorname{cp}(n)$. That is, $\varphi(n)$ is the number of integers from 1 to $n-1$ that are coprime to n.

Prime numbers are coprime to every smaller natural number. Therefore we see that $\varphi(2) = 1$, $\varphi(7) = 6$, and, in general, $\varphi(p) = p-1$ for every prime number p. Otherwise we need to be more careful; for example $\operatorname{cp}(4) = \{1, 3\}$, $\operatorname{cp}(6) = \{1, 5\}$, $\operatorname{cp}(8) = \{1, 3, 5, 7\}$, and $\operatorname{cp}(9) = \{1, 2, 4, 5, 7, 8\}$. Thus $\varphi(4) = 2$, $\varphi(6) = 2$, $\varphi(8) = 4$, and $\varphi(9) = 6$. Exercise 3.2.17 describes a general method of computing $\varphi(n)$ from the prime decomposition of n.

$\odot$	1	3	5	9	11	13
1	1	3	5	9	11	13
9	9	13	3	11	1	5
11	11	5	13	1	9	3

$\odot$	1	3	5	9	11	13
1	1	3	5	9	11	13
13	13	11	9	5	3	1

Figure 3.4. Idea for the proof of the Fermat-Euler Theorem.

3.2.7. Theorem (The Fermat-Euler Theorem)**.**
Let $n \geq 2$. Then $a^{\varphi(n)} \equiv 1 \pmod{n}$ for every integer a coprime to n. In particular, $\mathrm{ord}_n(a)$ divides $\varphi(n)$.

If $n = p$ is prime, then $\varphi(n) = p - 1$. So if $a \in \mathbb{Z}$ is not divisible by p, then the theorem tells us that a^{p-1} is congruent to 1 modulo p. Hence the Fermat-Euler Theorem contains Fermat's Little Theorem as a special case.

How might we prove the Fermat-Euler Theorem? To develop an idea, consider the case $n = 14$ and $a = 9$. The powers of a modulo n are given by

$$1, \quad 9, \quad 11, \quad 1, \quad 9, \quad 11, \quad \ldots.$$

Let A denote the set of these powers, i.e. $A = \{1, 9, 11\}$, and let us see what happens when we multiply all numbers in A with the same element of $\mathrm{cp}(n)$ (modulo n); see Figure 3.4. Whichever element we choose, we always obtain either the three numbers 1, 9, and 11 or the three numbers 3, 5, and 13.

Let us repeat the experiment for $a = 13$. Here $A = \{1, 13\}$, and multiplications give us three different classes of elements: 1 and 13, 3 and 11, as well as 5 and 9.

This gives us an idea for a proof of the Fermat-Euler Theorem. Suppose that, as in our example, the set $\mathrm{cp}(n)$ can always be subdivided into different subsets that all have the same size. Since A has exactly $\mathrm{ord}_n(a)$ elements (Exercise 3.2.13) and the number of elements of $\mathrm{cp}(n)$ is exactly $\varphi(n)$, this would imply that $\varphi(n)$ is divisible by $\mathrm{ord}_n(a)$, as claimed! We now work out the details of this idea.

Proof of the Fermat-Euler Theorem. What follows is perhaps the most complex proof in the first part of the book; we encourage the reader to fix an example, e.g. $n = 13$ and $a = 3$, and work out what each step of the proof means in this case. A simpler and slightly miraculous alternative proof is given as Exercise 3.2.25. Our approach has the advantage of generalizing to other situations.

As discussed above, we set

$$A := \{a^k \bmod n : k \geq 0\}. \tag{3.2.8}$$

(A reminder: $M \bmod n$ denotes the remainder of M when dividing by n.) To begin, we make two observations:

(I) If $b, c \in A$, then $b \cdot c \bmod n$ is also an element of A because the product of two powers of a is itself a power of a.

(II) For all $b \in A$, the set A contains an inverse of b modulo n. Indeed, we know that b and n are coprime, so by Lemma 3.2.1, there is a number $\ell \geq 2$ such that $b^\ell \equiv 1 \pmod n$. It follows from (I) that the power $c := b^{\ell-1} \bmod n$ is also an element of A, and we have $c \cdot b \equiv b^\ell \equiv 1 \pmod n$. Thus c is an inverse of b modulo n with $c \in A$, as desired.

(In the remainder of the proof we use only the properties (I) and (II), rather than the definition of A in (3.2.8). Since we were able to deduce (II) from (I), we are actually proving the theorem at the same time for all non-empty sets $A \subseteq \mathrm{cp}(n)$ that satisfy property (I). This is **Lagrange's Theorem** and is stated below.)

As above we now ask ourselves what happens when multiplying all elements of A with a fixed number $k \in \mathrm{cp}(n)$. In other words, we study the so-called **cosets** of A:

$$kA := \{kx \bmod n : x \in A\}.$$

Since $1 \in A$, every element of $\mathrm{cp}(n)$ is contained in at least one coset. We would like to show that the cosets subdivide $\mathrm{cp}(n)$ into subsets that all have the same number of elements. More precisely, we show:

(a) For all $k \in \mathrm{cp}(n)$, the set kA has exactly $\#A$ elements.

(b) If $k, l \in \mathrm{cp}(n)$ are such that kA and lA have a common element, then $kA = lA$.

By definition, the set kA has at most as many elements as A. Now let us assume, by contradiction, that kA contains *fewer* elements than A. Then there exist two different numbers $x, y \in A$ such that $kx \equiv ky \pmod{n}$. Since k and n are coprime, the cancellation rule (Corollary 3.1.6) tells us that $x \equiv y$ modulo n. But this is a contradiction because $x \neq y$ and A contains only elements from 1 to $n-1$. This proves (a).

If the sets kA and lA have a common element, then there are x and y in A such that

$$kx \equiv ly \pmod{n}. \tag{3.2.9}$$

Using (II), we can find an inverse x' of x modulo n in A. Multiplying the congruence (3.2.9) by x' on both sides, we see that

$$k = k \cdot 1 \equiv k \cdot xx' = kx \cdot x' \equiv ly \cdot x' = l(y \cdot x') \pmod{n}.$$

By (I), we see that $y \cdot x' \mod n \in A$, and hence $k \in lA$. It follows, again using (I), that $kz \bmod n \in lA$ for all $z \in A$. So we have proved that $kA \subseteq lA$. Exchanging the roles of k and l in this argument, we see that also $lA \subseteq kA$. This proves (b).

Now let m be the number of different sets that can occur as cosets. That is, there are $k_1, \dots, k_m \in \operatorname{cp}(n)$ such that $k_iA \neq k_jA$ for different i and j and furthermore $\operatorname{cp}(n) = k_1A \cup k_2A \cup \dots \cup k_mA$. From (b), we also know that no element of $\operatorname{cp}(n)$ is contained in more than one of the sets $k_1A, \dots, k_mA$, and hence

$$\varphi(n) = \#\operatorname{cp}(n) = \#k_1A + \dots + \#k_mA.$$

By (a), it follows that $\varphi(n) = m \cdot \#A = m \cdot \operatorname{ord}_n(a)$. The proof of the theorem is complete. ■

As we noted during the proof, we have actually established a more general fact, which will be useful in the next section:

3.2.10. Theorem (Lagrange's Theorem)**.**
Let $n \geq 2$ be a natural number. Let $A \subseteq \operatorname{cp}(n)$ be a non-empty set such that for any two (not necessarily different) elements k and l of A, the product $k \cdot l \bmod n$ also belongs to A.

Then the number $\#A$ of elements of A is a divisor of $\varphi(n)$.

Example. For $n = 15$, the set $A = \{1, 4, 11, 14\}$ satisfies the requirements of Lagrange's Theorem (Exercise 3.2.21), and, indeed, $\#A = 4$ divides $\varphi(15) = 8$.

Exercises.

3.2.11. Exercise. Compute the order of:
(a) 5 modulo 12; (b) 7 modulo 15; (c) 13 modulo 15.

3.2.12. Exercise. Let $n \geq 2$ and suppose that $a \in \mathbb{Z}$ is coprime to n and that b is inverse to a modulo n. Show that $\operatorname{ord}_n(a) = \operatorname{ord}_n(b)$.

3.2.13. Exercise (!). Let $n, a \in \mathbb{Z}$, $n \geq 2$, and suppose that a is coprime to n. Also let k be the order of a modulo n.

(a) Prove the following: if $b_1, b_2 \in \mathbb{N}_0$ are such that $b_1 \equiv b_2 \pmod{k}$, then $a^{b_1} \equiv a^{b_2} \pmod{n}$. (*Hint:* We may suppose that $b_2 \geq b_1$ and write $b_2 - b_1 = s \cdot k$ with $s \geq 0$. Then the proof is similar to the other parts of Lemma 3.2.1.)

(b) Consider the set $A := \{a^j \bmod n : j \geq 0\}$ of the remainders modulo n of all powers of a. Prove that

$$A = \{1\,,\ a \bmod n\,,\ a^2 \bmod n\,,\ \dots\,,\ a^{k-1} \bmod n\}.$$

(c) Deduce that A contains exactly $k = \operatorname{ord}_n(a)$ elements.

3.2.14. Exercise. Let p be an odd prime and m a natural number. Prove the following: if $m^2 + 1$ is divisible by p, then $p - 1$ is divisible by 4.

3.2.15. Exercise. Let p be an odd prime. Prove the following: if a is a natural number, then $a^p - a$ is a multiple of $2p$.

3.2.16. Exercise. Let p be prime. Show that $(a + b)^p \equiv a^p + b^p \pmod{p}$ for all natural numbers a and b.

3.2.17. Exercise (!). Let $n, m \in \mathbb{N}$. Prove the following:

(a) If n and m are coprime, then $\varphi(n \cdot m) = \varphi(n) \cdot \varphi(m)$.

(b) If p is prime and $k \in \mathbb{N}$, then $\varphi(p^k) = (p - 1) \cdot p^{k-1}$.

Use these rules to compute $\varphi(10)$, $\varphi(50)$, and $\varphi(180)$. Show also that if $n > 2$, then $\varphi(n)$ is even.

3.2.18. Exercise. Let $n \geq 2$ be a natural number. Prove that

$$n = \sum_{k|n} \varphi(k),$$

where the sum is taken over all natural numbers k that divide n. Here we use the convention that $\varphi(1) := 1$. (*Hint:* Let k be a divisor of n. How many natural numbers $\leq n$ are there such that $\gcd(a,n) = k$? Also use the fact that $\sum_{k|n} \varphi(k) = \sum_{k|n} \varphi(n/k)$.)

3.2.19. Exercise. Why is $m^5 \equiv m \pmod{10}$ for every integer m?

3.2.20. Exercise. Show that $m^7 - m$ is a multiple of 42 for all natural numbers m.

Show more generally that the Fermat-Euler Theorem can be strengthened as follows: suppose that $r, s \geq 2$ are coprime integers and set $n := r \cdot s$. If $a \in \mathbb{Z}$ is coprime to n, then

$$a^{\operatorname{lcm}(\varphi(r),\varphi(s))} \equiv 1 \pmod{n}.$$

(*Hint:* Use the Chinese Remainder Theorem.)

3.2.21. Exercise. Show that (e.g. by writing down a multiplication table for A), for $n = 15$, the set $A = \{1, 4, 11, 14\}$ satisfies the hypothesis of Lagrange's Theorem.

Further Exercises and Comments.

3.2.22. Exercise. Let $n = p^k$, where p is prime and $k \geq 2$. Prove that there exists a number $a \in \operatorname{cp}(n)$ with $\operatorname{ord}_n(a) = p$. Show that this is true also for every number n that is divisible by p^2. (*Hint:* For the first part, consider the number $a = p^{k-1} + 1$ and use Lemmas 3.2.4 and 3.2.1. For the second claim use the first part and the Chinese Remainder Theorem.)

3.2.23. Exercise. Figure 3.2 suggests another question: for which numbers $n \geq 2$ is there a coprime number a such that $\operatorname{ord}_n(a) = \varphi(n)$? Such a number is called a **primitive root modulo** n. Apart from $n = 8$, all the numbers in our table have a primitive root. In general, the following are true:

(a) If $n = 2^k$ with $k \geq 3$, then there is no primitive root modulo n.

(b) If $n = r \cdot s$, where $r > 2$ and $s > 2$ are coprime, then there is no primitive root modulo n.

(c) Otherwise (in particular when n is prime), there are primitive roots modulo n. (See Exercise 6.4.9.)

Prove (a) and (b). (*Hint:* Show that there are at least four numbers in $\operatorname{cp}(n)$ whose square is congruent to 1 modulo n. Deduce that these numbers cannot all appear as powers of the same number a. For (b), it is possible to alternatively use Exercise 3.2.20.)

3.2.24. Euler introduced the function named after him in 1760. It seems that the use of the symbol φ for this function was first introduced by Gauss in 1801 and that the word “totient” was coined by Sylvester in 1879. Sometimes the function is instead referred to simply as **Euler’s phi function**.

3.2.25. Exercise. Prove the Fermat-Euler Theorem by working through the following steps:

Let $T = \{b_1, b_2, \ldots, b_{\varphi(n)}\}$ be a reduced set of residues (RSR) modulo n (as defined in Comment 3.1.22). For example, we can use $T = \mathrm{cp}(n)$. By Exercise 3.1.23, the set $aT = \{a \cdot b_1, a \cdot b_2, \ldots, a \cdot b_{\varphi(n)}\}$ is also an RSR modulo n. Prove the following:

(a) $b_1 \cdot b_2 \cdots b_{\varphi(n)} \equiv (a \cdot b_1) \cdot (a \cdot b_2) \cdots (a \cdot b_{\varphi(n)}) \pmod{n}$.

(b) $b_1 \cdot b_2 \cdots b_{\varphi(n)} \equiv (a^{\varphi(n)}) \cdot (b_1 \cdot b_2 \cdots b_{\varphi(n)}) \pmod{n}$.

(c) $1 \equiv a^{\varphi(n)} \pmod{n}$.

3.2.26. There are much more general versions of Lagrange’s Theorem (Theorem 3.2.10) than the one we stated and proved. Indeed, in the proof we only need to know that the product of two numbers modulo n again belongs to $\mathrm{cp}(n)$, that the number 1 is in $\mathrm{cp}(n)$, that multiplication modulo n is associative, and that every element of $\mathrm{cp}(n)$ has an inverse modulo n.

A non-empty set on which there is a binary operation with these properties is called a **group**. **Group theory** is a large mathematical field with many connections to other areas of mathematics and even e.g. to physics and computer science. A simple example of a group is given by the symmetries of an equilateral triangle, with the composition of symmetries as the binary operation. The set of all possible transformations of a Rubik’s Cube can also be studied using the methods of group theory. For further reading we recommend the introductory textbook [**KS**].

3.3. A first primality test

Theorem 3.2.2 provides us with a property of prime numbers that can easily be tested. Indeed, if $n \geq 2$ is any natural number, then we can pick a number a that is coprime to n and such that $1 \leq a < n$ and check whether or not the congruence

$$a^{n-1} \equiv 1 \pmod{n} \tag{3.3.1}$$

holds. Such a number a is called a **base**. We remind the reader that powers modulo n can be computed efficiently using the “divide and conquer” principle; see Exercise 3.1.14. If the congruence (3.3.1) does not hold, then we know by Fermat's Little Theorem that n must be composite! Hence we can formulate the following algorithm, which is called the **Fermat test** for obvious reasons:

```
ALGORITHM FERMAT-TEST
Input: A number $n \geq 2$.
  1. Choose (e.g. at random) a natural number $a$ from 1 to $n-1$.
  2. If $\gcd(a,n) \neq 1$, then output "$n$ is composite".
  3. Otherwise compute $a^{n-1} \bmod n$ using the algorithm from Exercise 3.1.14.
  4. If the number computed in the previous step is equal to 1, then output "$n$ may be prime"; otherwise output "$n$ is composite".
```

What is happening here? Clearly there are two cases to consider: either n is prime or not. If we input a prime number n, then every smaller number $a \geq 1$ is coprime to n, so we pass on to Step **3**. Fermat's Little Theorem guarantees that $a^{n-1} \equiv 1 \pmod{n}$, so primes are always correctly identified as such. If the input n is composite, then the situation is not quite as simple. However, examples suggest that the Fermat test works well for small numbers, even if we fix a basis such as $a = 2$ (Exercise 3.3.4), and that it can quickly recognize even extremely large composite numbers (Exercise 3.3.6).

So we are faced with the following question: are there numbers $a < n$ with $a \neq 1$ such that the test does *not* recognize n to be composite? I.e. could it happen that $a \in \mathrm{cp}(n)$ with $a > 1$ satisfies $a^{n-1} \equiv 1 \pmod{n}$ even though n is not prime?

Unfortunately this phenomenon does indeed occur – there are numbers that "pretend" to be prime. In this case, we call n a **pseudo-prime** to base a. For example, every odd composite number n is a pseudo-prime to base $a = n-1$. But there are other examples; one is $11^{14} = (11^2)^7 = 121^7 \equiv 1^7 = 1 \pmod{15}$, although 15 is not prime. So if we input $n = 15$ and happen to pick the basis $a = 11$, then the Fermat test does not recognize 15 as a composite number.

We can still hope that at least there are not *too many* bases with respect to which a composite number n is a pseudo-prime. That is, we ask how large the set

$$A := \{a \in \mathrm{cp}(n) : a^{n-1} \equiv 1 \pmod{n}\} \tag{3.3.2}$$

can be. An interesting observation is that A is closed under multiplication modulo n: if $a, b \in A$, then

$$(a \cdot b)^{n-1} = a^{n-1} \cdot b^{n-1} \equiv 1 \cdot 1 = 1 \pmod{n},$$

so $a \cdot b \bmod n$ is again an element of A. This means that A satisfies the assumptions of Lagrange's Theorem that we discussed in the preceding chapter! We can thus show:

3.3.3. Lemma (Number of elements of A)**.**
Let n be a composite number and define A as in (3.3.2)*. If $A \neq \mathrm{cp}(n)$, then A contains at most $\varphi(n)/2$ elements.*

Proof. By Lagrange's Theorem, $\#A$ divides $\varphi(n)$; that is, there is $k \in \mathbb{N}$ such that $\varphi(n) = k \cdot \#A$. By assumption, we have $\#A < \varphi(n)$, and hence $k \geq 2$ and $\#A = \varphi(n)/k \leq \varphi(n)/2$. ■

This is great news: if there is *any* suitable base a, then the probability of finding one is at least $1/2$. If we could show that, for all composite numbers n, there exists a base $a \in \mathrm{cp}(n)$ such that n is not a pseudo-prime for this base, then this would prove that the Fermat test is an efficient Monte Carlo algorithm for COMPOSITES.

Unfortunately for us, this is false. The counterexamples are called **Carmichael numbers**; the smallest one is $561 = 3 \cdot 11 \cdot 17$. It is even possible to show that there are infinitely many Carmichael numbers, and our test would mistake all of these for primes. However, we have nonetheless made significant progress: on the one hand, our test never mistakes a prime number for a composite, and on the other hand, we found our first method that can detect even very large composite numbers (unless they happen to be Carmichael numbers, which are in fact quite rare). In Section 4.5, we will improve the Fermat test in such a way that even Carmichael numbers do not cause problems anymore.

Exercises.

3.3.4. Exercise. Carry out the Fermat test (by hand or with a calculator) to base $a = 2$ for the numbers $n = 9, 21, 25, 27, 33, 35$. Are these recognized to be composite?

3.3.5. Exercise (P). Implement the Fermat test in a common programming language and apply it to the numbers 1 726 374 899 084 624 209 and 6 641 819 896 288 796 729. Are they recognized to be composite?

3.3.6. Exercise (P). Apply the Fermat test (e.g. to base $a = 2$) to the composite number RSA-2048 given in the introduction to this book. (To do so, use a calculator that can carry out modular arithmetic for numbers with arbitrarily many digits; see Exercise 3.1.15.)

Further Exercises and Comments.

3.3.7. Exercise. Show that 561 really is a Carmichael number, as claimed. (*Hint:* Exercise 3.2.20.)

3.3.8. Exercise. Show that all Carmichael numbers are odd.

3.3.9. Exercise. Show that a Carmichael number cannot contain any prime factor more than once. Hence if n is a natural number and if p is a prime such that n is divisible by p^2, then n is not a Carmichael number. (*Hint:* Exercise 3.2.22.)

3.3.10. Exercise. If p is prime, then there is some number a such that $\mathrm{ord}_p(a) = p - 1$ (Exercise 6.4.9; see also Exercise 3.2.23). Use this fact and the Chinese Remainder Theorem to show: if $q \in \mathbb{N}$ with $1 < q < p$, then $n := p \cdot q$ is not a Carmichael number. (See also the proof of Theorem 4.5.4.)

3.3.11. Exercise. Using the preceding two exercises, show that every Carmichael number must have at least three different prime divisors.

3.4. Polynomials

Polynomials and polynomial long division are often covered in high school. Since they are of crucial importance for the second part of our book, we would like to use this section to recall their elementary properties, collect rules of arithmetic, and gain some more practice, so that later on the reader will feel comfortable performing calculations with polynomials. The idea is to treat polynomials just as if they were numbers. At the end of the section we shall take this idea further and develop the concepts of **modular arithmetic with respect to a polynomial** (analogously to modular arithmetic for integers) and of **irreducible polynomials** (analogs of prime numbers).

3.4.1. Definition.
An (integer/rational) **polynomial** P is a sum of the form

$$P = a_n X^n + a_{n-1} X^{n-1} + \cdots + a_1 X + a_0,$$

where $n \in \mathbb{N}_0$ and $a_0, ..., a_n$ are (integer/rational) numbers; they are called the **coefficients**. The **degree** of P, denoted by $\deg P$, is the highest exponent d for which the coefficient of X^d is nonzero. This coefficient is then called the **leading coefficient** of P, and the polynomial is **monic** if its leading coefficient is 1.

The **zero polynomial**, for which all coefficients are equal to zero, has degree $-\infty$ by definition; it is the only polynomial that does not have a leading coefficient.

Examples. $3X^2-1$ is an integer polynomial of degree 2. Its coefficients are $a_2 = 3$, $a_1 = 0$, and $a_0 = -1$; in particular the leading coefficient is 3. Polynomials of degree ≤ 0 are called **constant polynomials**. For example, -5 is a constant integer polynomial.

In the remainder of the book, we are really only interested in integer polynomials; if we write simply “polynomial”, we thus always assume that the coefficients are integers. In this section, and only here, we sometimes also allow rational coefficients, in particular when discussing division with remainder (see below). Of course it is also possible to study polynomials with real or even more general coefficients; see Comment 3.4.23. Arithmetic with polynomials, such as addition, subtraction, and multiplication, is intuitive and straightforward, and we refrain from giving an unnecessarily formal treatment. However, let us make a few clarifying remarks to avoid confusion.

Substitution into polynomials. We can substitute every integer (or also rational/real number) x into a polynomial and then write $P(x)$ for the corresponding value. If $P = 3X^2 - 1$, for example, then we have $P(1) = 3 \cdot 1^2 - 1 = 2$. A number x for which $P(x) = 0$ is called a **zero** of the polynomial P.

When writing down a polynomial, the name of the variable does not really matter. Usually we call it X, but sometimes it is useful to have other names for variables at our disposal in this case we always use Y or Z. Then we refer to polynomials **in** X, resp. in Y or in Z. For example, $Y^3 - 1$ is a polynomial in Y and $2Z^4 + 3Z$ is a polynomial in Z.

Equality of polynomials. By definition, two polynomials are **equal** if they have the same coefficients. Note that, when writing down a polynomial, we usually leave out terms where the coefficient is zero, as we did in the examples above. For instance, we have

$$3X^2 - 1 = 0 \cdot X^3 + 3 \cdot X^2 + 0 \cdot X + (-1).$$

Addition and multiplication. If P and Q are polynomials, then their sum $P + Q$ is again a polynomial (combining the terms that correspond to the same power of X). For example, suppose that $P = 2X^4 - X^2 + 3X$ and $Q = -2X^4 + X^3 - 5X - 1$:

$$\begin{aligned} P + Q &= (2X^4 - X^2 + 3X) + (-2X^4 + X^3 - 5X - 1) \\ &= (2-2)X^4 + X^3 - X^2 + (3-5)X - 1 = X^3 - X^2 - 2X - 1. \end{aligned}$$

We see at once that the degree of $P + Q$ can be at most as large as the higher of the two degrees of P and Q. In our example, the degree even decreased, because the two terms involving X^4 canceled out.

In the same way, we can compute the product $P \cdot Q$, which is again a polynomial. For $P = X^2 + X - 2$ and $Q = X^3 - 3X$ this looks as follows:

$$\begin{aligned} P \cdot Q &= (X^2 + X - 2) \cdot (X^3 - 3X) \\ &= X^5 + X^4 - 2X^3 - 3X^3 - 3X^2 + 6X \\ &= X^5 + X^4 - 5X^3 - 3X^2 + 6X. \end{aligned}$$

The leading coefficient of $P \cdot Q$ is the product of the leading coefficients of P and Q. In particular, $\deg(P \cdot Q) = \deg P + \deg Q$.

In practice, it is often sensible not to multiply out a polynomial that is given as a product. For example, the form

$$P = (X+1)^{10}$$

is much more practical and easier to understand than the expansion

$$\begin{aligned} P = {} & X^{10} + 10X^9 + 45X^8 + 120X^7 + 210X^6 \\ & + 252X^5 + 210X^4 + 120X^3 + 45X^2 + 10X + 1 \end{aligned}$$

according to the binomial theorem!

We have seen that we can add, subtract, and multiply polynomials. It is natural to ask about division. If P is not the zero polynomial, then we can use **polynomial long division** to divide a polynomial Q with remainder by P. (This means that we are looking for T and R such that $\deg R < \deg Q$ and $Q = T \cdot P + R$.) For completeness, let us explain this method by showing how to divide $Q = 12X^2 - 11X - 1$ by $P = X - 2$. The idea is to determine the coefficients of T one by one, beginning with the leading coefficient. So we start by looking at the first term on both sides and see that $12X \cdot X$ will give us $12X^2$, so the highest term of T is $12X$. We compute $12X \cdot P = 12X^2 - 24X$ and subtract this from the original polynomial Q: $12X^2 - 11X - 1 - (12X^2 - 24X) = 13X - 1$. Now we do the same for $13X - 1$ and see that $13 \cdot X = 13X$. So we note 13 as the second term of the result, multiply again $13 \cdot P = 13X - 26$, and obtain the remainder $(13X - 1) - (13X - 26) = 25$. This remainder has smaller degree than P, and hence we are done, having obtained $T = 12X + 13$ and $R = 25$. The usual way to write down the calculations is as follows:

$$\begin{array}{lrrl}
\big(& 12X^2 - 11X & -1\big) & : \big(X-2\big) = 12X + 13; \quad \textbf{remainder: } 25. \\
 & -12X^2 + 24X & & \\ \cline{2-2}
 & 13X & -1 & \\
 & -13X & +26 & \\ \cline{2-3}
 & & 25 &
\end{array}$$

If, instead, we had divided $12X^2 - 11X - 26$ by $X - 2$, we would have been left with remainder zero; this means that $X - 2$ is a **divisor** of $12X^2 - 11X - 26$:

$$\begin{array}{l}
\bigl(\quad 12X^2 - 11X - 26\bigr) : \bigl(X - 2\bigr) = 12X + 13 \\
\underline{\ -12X^2 + 24X\ } \\
\qquad\qquad\quad 13X - 26 \\
\qquad\qquad \underline{-13X + 26} \\
\qquad\qquad\qquad\qquad 0
\end{array}$$

We also note that the polynomials T and R in our examples have integer coefficients because P is monic, and hence in every step the division by the leading coefficient of P yields an integer. Overall we have the following theorem:

3.4.2. Theorem (Division with remainder)**.**
Let P and Q be rational polynomials. Then there are rational polynomials T and R such that $\deg R < \deg P$ *and*

$$Q = T \cdot P + R,$$

*and these are uniquely determined by P and Q. We say: P **divides** Q **with remainder** R. If P and Q are integer polynomials and P is monic, then T and R also have integer coefficients.*

Proof. The theorem follows from the method of polynomial long division, which works for all polynomials P and Q as stated. Uniqueness also follows immediately because, in every step, we have only one possibility for picking the corresponding coefficient of T. Alternatively we can prove uniqueness directly. Indeed, if $Q = T_1 \cdot P + R_1$ and $Q = T_2 \cdot P + R_2$ are two representations of Q as above, then

$$(T_1 - T_2) \cdot P + (R_1 - R_2) = 0.$$

Since R_1 and R_2 have smaller degree than P, we must also have $\deg(R_1 - R_2) < \deg P$. But then the first term $(T_1 - T_2) \cdot P$ in the sum also necessarily has smaller degree than P. Thus

$$\deg P > \deg\bigl((T_1 - T_2) \cdot P\bigr) = \deg(T_1 - T_2) + \deg P.$$

This is only possible if $\deg(T_1 - T_2) < 0$; i.e. $T_1 - T_2$ is the zero polynomial. This means that $T_1 = T_2$, which also implies $R_1 = R_2$ as claimed. ■

3.4.3. Definition (Divisors of polynomials)**.**
Let P and Q be rational polynomials. If there is a rational polynomial T such that $Q = T \cdot P$, then P is called a **divisor of Q over $\mathbb{Q}$**. If additionally $\deg P \neq 0$ and $\deg T \neq 0$, then we call P and T **non-trivial divisors of Q over $\mathbb{Q}$**.

If P, Q, and T are integer polynomials with $Q = T \cdot P$, then we analogously call P and T **divisors of Q over $\mathbb{Z}$**. If neither P nor T is equal to 1 or -1, then P and T are called **non-trivial divisors of Q over $\mathbb{Z}$**.

Remark. Even for integer polynomials, divisibility over $\mathbb{Q}$ can mean something different than divisibility over $\mathbb{Z}$. For example, $2X + 2$ is a divisor of $3X + 3$ over $\mathbb{Q}$ because $3X + 3 = \frac{3}{2} \cdot (2X + 2)$. On the other hand, $2X + 2$ is *not* a divisor of $3X + 3$ over $\mathbb{Z}$. Similarly it is true that every rational polynomial P of degree 1 (that is, $P = a$ with $a \in \mathbb{Q} \setminus \{0\}$) divides every polynomial over $\mathbb{Q}$. When dividing over $\mathbb{Z}$, this is true only for the constant polynomials 1 and -1; this is the reason for the different definitions of non-trivial divisors.

For us, divisibility over $\mathbb{Q}$ is more important since it introduces us to ideas required in the next section. The connection between both notions of divisibility is explored in Exercise 3.4.20.

An important application of division with remainder is a result that the reader may be familiar with from secondary school:

3.4.4. Theorem (Linear factors)**.**
Let P be a rational polynomial and let $a \in \mathbb{Q}$. Then a is a zero of P if and only if the ***linear factor*** *$X - a$ is a divisor of P over $\mathbb{Q}$.*

Proof. If $X - a$ is a divisor of P, then we can write $P = Q \cdot (X - a)$ for a suitable rational polynomial Q and we see that indeed

$$P(a) = Q(a) \cdot (a - a) = Q(a) \cdot 0 = 0.$$

Conversely, let a be a zero of P. We divide P by $X - a$ with remainder: $P = T \cdot (X - a) + R$, where $\deg R < \deg(X - a) = 1$. We must show that $R = 0$.

As $\deg R < 1$, we already know that R is a constant polynomial, so $P = T \cdot (X - a) + b$ for some $b \in \mathbb{Q}$. We only need to show that $b = 0$, and this follows from the fact that a is a zero of P:

$$0 = P(a) = T(a) \cdot (a - a) + b = T(a) \cdot 0 + b = 0 + b = b. \qquad \blacksquare$$

Example. Let $P := 2X^3 + X^2 - 5X + 2$. Then -2 is a zero of P since $P(-2) = 2 \cdot (-8) + 4 - 5 \cdot (-2) + 2 = -16 + 4 + 10 + 2 = 0$, and indeed $P = (2X^2 - 3X + 1) \cdot (X + 2)$, so $(X + 2)$ is a divisor of P.

3.4.5. Corollary (Number of zeros)**.**
A rational polynomial of degree $d \geq 0$ has at most d zeros in $\mathbb{Q}$.

Proof. The claim follows from the preceding theorem by induction. A polynomial of degree 0 is, by definition, a non-zero constant. Hence it has no zeros, which establishes the basis of the induction.

Now let P be a rational polynomial of degree $d \geq 1$. If there are no zeros, then we are done, so we can assume that P has at least one zero $a \in \mathbb{Q}$. By Theorem 3.4.4 we can write $P = (X - a) \cdot T$ for some rational polynomial T of degree $d - 1$. Then, every zero of P different from a must also be a zero of T. By the induction hypothesis, T has at most $d - 1$ zeros in $\mathbb{Q}$. Consequently, P has at most d zeros, as claimed. ■

Modular arithmetic for polynomials and irreducibility. Let P, Q, and H be rational polynomials with $\deg H \geq 1$. Similarly to modular arithmetic in $\mathbb{Z}$, we write

$$P \equiv Q \pmod{H}$$

if both polynomials have the same remainder when divided by H. Again, we can do elementary arithmetic modulo H as usual, e.g.

$$\begin{aligned}(2X^2 - 3X + 1) \cdot (X + 2) &\equiv (-3X - 1) \cdot (X + 2) \\ &= -3X^2 - 7X - 2 \\ &\equiv -7X + 1 \qquad \pmod{X^2 + 1}.\end{aligned}$$

In $\mathbb{Z}$, we saw that arithmetic modulo a prime number has particularly nice properties, so we now define a property of polynomials that is analogous to primality.

3.4.6. Definition (Irreducibility)**.**
A non-constant integer polynomial H is called **irreducible over** $\mathbb{Q}$, resp. **over** $\mathbb{Z}$, if it does not have any non-trivial divisors over $\mathbb{Q}$, resp. over $\mathbb{Z}$.

Examples. Every monic polynomial of degree 1 is irreducible both over $\mathbb{Q}$ and over $\mathbb{Z}$, by definition. The polynomial $X^2 - 2$ is also irreducible over $\mathbb{Q}$: the only possible non-trivial divisors would be polynomials of degree 1 and thus would have the form $aX - b$ with $a, b \in \mathbb{Q}$. In particular, $\frac{b}{a}$ would have to be a zero, but $X^2 - 2$ does not have any rational zeros by Theorem 1.1.2. For the same reason, $X^2 - 2$ is irreducible over $\mathbb{Z}$. On the other hand, $2X^3 + X^2 - 5X + 2$ is not irreducible over $\mathbb{Q}$ and also not over $\mathbb{Z}$: we already saw in the example for Theorem 3.4.4 that it has the non-trivial divisor $X + 2$.

As a final example we note that $2X + 2$ is irreducible over $\mathbb{Q}$, but not over $\mathbb{Z}$: the constant polynomial 2 is a non-trivial divisor over $\mathbb{Z}$, but a trivial divisor over $\mathbb{Q}$. However, a polynomial that is irreducible over $\mathbb{Z}$ is always also irreducible over $\mathbb{Q}$; see Exercise 3.4.21.

The following theorem tells us that irreducible polynomials really do behave similarly to prime numbers. (We state and prove it only over $\mathbb{Q}$, but we will see in Exercise 3.4.21 that a corresponding statement is true also over $\mathbb{Z}$.)

3.4.7. Theorem (Irreducible divisors of a product)**.**
Let H be a non-constant polynomial that is irreducible over $\mathbb{Q}$. If Q_1 and Q_2 are rational polynomials such that H is a divisor of $Q_1 \cdot Q_2$ over $\mathbb{Q}$, then H must divide one of the factors Q_1 and Q_2.

Proof. We follow essentially the same idea as in the proof of the Fundamental Theorem of Arithmetic. Let R_1 and R_2 be the remainders of Q_1 and Q_2 when dividing by H, say $Q_1 = T_1 \cdot H + R_1$ and

$Q_2 = T_2 \cdot H + R_2$. Multiplying out and combining terms, we see that

$$Q_1 \cdot Q_2 = H \cdot (T_1 \cdot T_2 \cdot H + T_1 \cdot R_2 + T_2 \cdot R_1) + R_1 \cdot R_2.$$

So $R := R_1 \cdot R_2$ is also divisible by H. We must show that R_1 or R_2 is equal to zero; that is, we should show that R is the zero polynomial. We do this by using the method of the smallest counterexample. If the claim is false, then we can choose H with the lowest possible degree; hence we may suppose that the theorem is true for all irreducible polynomials of smaller degree. In addition, we suppose that the degree of R is as small as possible with our choice of H.

As R is divisible by H, there exists a rational polynomial T such that $R = T \cdot H$. We have $\deg R = \deg R_1 + \deg R_2 < 2 \cdot \deg H$, so it follows that $\deg T < \deg H$. If T is constant, then R_1 and R_2 are non-trivial divisors of H over $\mathbb{Q}$, which is impossible because H is irreducible. So we know that $\deg T \geq 1$.

Thus T has some irreducible factor I over $\mathbb{Q}$ (see Exercise 3.4.16). We know that $1 \leq \deg I \leq \deg T < \deg H$. So, by choice of H as a counterexample of smallest degree, the theorem is true for I. As I divides the product $R = R_1 \cdot R_2$, we see that I divides one of the two factors, say R_1.

If we now write $R_1 = \tilde{R}_1 \cdot I$, then H still divides the product $\tilde{R} := \tilde{R}_1 \cdot R_2$ over $\mathbb{Q}$ but does not divide either of the factors $\tilde{R}_1$ or R_2. (This is because their degree is smaller than that of H and they are non-zero by assumption.) But the degree of $\tilde{R}_1$ is strictly smaller than that of R_1, and thus also $\deg \tilde{R} < \deg R$. This contradicts our choice of R as having minimal degree, and the proof is complete. ■

3.4.8. Corollary (Zero divisors)**.**
Let H be a rational polynomial that is non-constant and irreducible over $\mathbb{Q}$. Let Q_1 and Q_2 be rational polynomials. If

$$Q_1 \cdot Q_2 \equiv 0 \pmod{H},$$

then $Q_1 \equiv 0 \pmod{H}$ or $Q_2 \equiv 0 \pmod{H}$.

Proof. The reader is invited to verify that this is nothing but a reformulation of the theorem we have just proved. ■

Polynomial zeros. Let P be a polynomial in Y and let Q be a polynomial in X. Then we can substitute Q for the variable Y in P and obtain $P(Q)$, which is a polynomial in X. This is easiest to understand using an example: if $P := Y^2 - 2$ and $Q := X - 3$, then

$$P(Q) = (X-3)^2 - 2 = X^2 - 6X + 9 - 2 = X^2 - 6X + 7. \tag{3.4.9}$$

This gets particularly interesting when we are doing modular arithmetic with respect to a polynomial. Indeed, if H is a non-constant monic polynomial in X, then it can happen that $P(Q)$ is divisible by H, and hence congruent to zero. This means that Q is, in some sense, a "zero" of the polynomial P modulo H.

3.4.10. Definition (Polynomial zeros)**.**
Let H be a non-constant rational polynomial, and let P and Q be rational polynomials such that

$$P(Q) \equiv 0 \pmod{H}.$$

Then Q is called a **polynomial zero** of P (modulo H).

Example. From (3.4.9) we know that $X - 3$ is a polynomial zero of $Y^2 - 2$ modulo $H := X^2 - 6X + 7$.

We already announced that calculations modulo an irreducible polynomial are particularly pleasant. We now see an example of this principle: if H is irreducible over $\mathbb{Q}$, then a polynomial can have at most as many polynomial zeros as indicated by its degree (where we consider two polynomial zeros to be the same if they are congruent modulo H).

3.4.11. Theorem (Number of polynomial zeros)**.**
Let H be a non-constant rational polynomial that is irreducible over $\mathbb{Q}$, and let P be a rational polynomial of degree $d \geq 0$.

Then P has at most d polynomial zeros modulo H that are pairwise different modulo H. (This means that there are at most d polynomials $Q_1, \ldots, Q_d$ that are pairwise incongruent modulo H and that satisfy $P(Q_j) \equiv 0 \pmod{H}$.)

Proof. We use the same idea as in Corollary 3.4.5, where we separated out a linear factor using polynomial long division.

We can do the same here if we think of P as a polynomial in Y *whose coefficients are polynomials in* X. (A more abstract point of view is as follows: we can allow any type of coefficients for polynomials, as long as we can add, subtract, and multiply them. Using this idea, both Theorem 3.4.11 and Corollary 3.4.5 are just special cases of the same statement; see Exercise 3.4.24.) We show: if

$$P := A_d Y^d + A_{d-1} Y^{d-1} + \cdots + A_1 Y + A_0$$

is a "polynomial" in Y whose coefficients A_i are themselves rational polynomials in X, then – up to congruence modulo H – there are at most d polynomials Q such that

$$P(Q) \equiv 0 \pmod{H}.$$

The proof is essentially the same as before. If Q has this property, then we can divide P by the polynomial $T := Y - Q$ with remainder. So we write $P = \tilde{P} \cdot T + R$. By choice of Q we have

$$R(Q) \equiv P(Q) \equiv 0 \pmod{H}.$$

Since R has degree 1 (as a polynomial in Y), we know that R must be constant in Y. So $R \equiv 0 \pmod{H}$ and thus

$$P \equiv \tilde{P} \cdot (Y - Q) \pmod{H}.$$

By Corollary 3.4.8, up to congruence modulo H the only polynomial zeros of P are Q and the polynomial zeros of $\tilde{P}$. The latter polynomial has degree at most $d - 1$, and the claim follows inductively. ■

Exercises.

3.4.12. Exercise. Use polynomial long division to compute

(a) $(X^4 - 1) : (X^2 - 1)$,
(b) $(X^5 + X^4 + X^3 + X^2 + X + 1) : (X^2 + X + 1)$, and
(c) $(2X^2 - X) : \left(X - \frac{1}{2}\right)$.

Also divide the polynomial $2X^2 + 3X + 5$ by $3X$ with remainder.

3.4.13. Exercise. Let P and Q be polynomials. Show that P and Q are equal if and only if $P(x) = Q(x)$ for all $x \in \mathbb{Z}$.

3.4.14. Exercise.

(a) Are the following polynomials irreducible?
  (i) $x^2 - 1$ (over $\mathbb{Z}$, $\mathbb{Q}$).
  (ii) $x^2 + 1$ (over $\mathbb{Q}$).
  (iii) $3x^4 + 2x^2 - 6x + 1$ (over $\mathbb{Z}$, $\mathbb{Q}$).
  (iv) $x^4 + 1$ (over $\mathbb{Z}$).

(b) Find polynomials P, Q with integer coefficients such that $P \cdot Q$ is divisible by $2X + 2$ over $\mathbb{Z}$, but neither P nor Q is divisible by $2X + 2$ over $\mathbb{Z}$.

3.4.15. Exercise. Prove that a polynomial of degree 2 is irreducible over $\mathbb{Q}$ if and only if it has no zeros in the rational numbers.

3.4.16. Exercise (!). Let P be a non-constant rational polynomial. Show that there is a polynomial H with integer coefficients such that H is irreducible over $\mathbb{Q}$ and divides P over $\mathbb{Q}$.

3.4.17. Exercise. We saw in (3.4.9) that $X - 3$ is a polynomial zero of $Y^2 - 2$ modulo $H := X^2 - 6X + 7$. Find (using, for example, a polynomial long division as in the proof of Theorem 3.4.11) a second polynomial zero.

3.4.18. Exercise. Find a polynomial Q of degree 2 and a polynomial H that is *reducible* (i.e. not irreducible) over $\mathbb{Q}$ such that Q has at least three polynomial zeros modulo H no two of which are congruent modulo H. Hence Theorem 3.4.11 is no longer true for reducible polynomials.

Further Exercises and Comments.

3.4.19. Exercise. Let p be prime and let P and Q be integer polynomials. Prove the following: if $P \cdot Q$ is divisible by p (i.e. all coefficients are divisible by p), then p divides P or Q. (*Hint:* Write $P = a_0 + a_1X + \cdots + a_nX^n$ and $Q = b_0 + b_1X + \cdots + b_mX^m$. If our claim is false, then consider the smallest numbers k and l such that a_k and b_l are not divisible by p. What can you say about the $(k + l)$-th coefficient of $P \cdot Q$?)

This suggests that prime numbers should be considered to be constant irreducible polynomials over $\mathbb{Z}$. (We previously defined irreducibility only for *non-constant* polynomials.) On the other hand, there are no constant irreducible polynomials over $\mathbb{Q}$, just as 1 is not a prime number.

3.4.20. Exercise. Let H be a non-constant integer polynomial whose coefficients do not all have some prime factor in common. Also let Q be an arbitrary integer polynomial. Prove the following: if H is a divisor of Q over $\mathbb{Q}$, then H is also a divisor of Q over $\mathbb{Z}$.

(*Hint:* Write $Q = T \cdot H$, where T is rational. Let k be the least common multiple of the denominators of the coefficients of T. Then $k \cdot T$ is an integer polynomial that is not divisible by any prime divisor of k. Furthermore, $k \cdot Q = (k \cdot T) \cdot H$. Now use Exercise 3.4.19 to see that we must have $k = 1$.)

3.4.21. Exercise. Show, using the preceding exercises, that a non-constant polynomial H is irreducible over $\mathbb{Z}$ if and only if it is irreducible over $\mathbb{Q}$ and its coefficients do not all have a common prime factor.

Deduce from this, using Theorem 3.4.7: if H is irreducible over $\mathbb{Z}$ and P and Q are integer polynomials such that H is a divisor of $P \cdot Q$ over $\mathbb{Q}$, then H is also a divisor of P or of Q over $\mathbb{Z}$.

3.4.22. The integers and polynomials with integer coefficients have many properties in common:

- The commutative, associative, and distributive laws hold for addition and multiplication.
- We can subtract any element from any other.
- In both systems there is a "zero" 0 and a "one" 1; i.e. for all (integers, resp. polynomials) a, we have $a + 0 = a$ and $a \cdot 1 = a$.
- There are no zero divisors; i.e. if $a \cdot b = 0$, then $a = 0$ or $b = 0$.

We follow the same idea as in the definition of *fields* in Comment 3.1.24 and call a system with these properties an **integral domain**. In particular, every field is an integral domain but not every integral domain is a field.

3.4.23. We restricted ourselves to studying integer and rational polynomials, but we could in principle allow polynomials whose coefficients belong to any field or integral domain. Notions such as divisibility, irreducibility, etc., can likewise be defined in a completely general manner. This point of view has the advantage that we do not have to prove the same statement several times for different number systems. The following exercises illustrate this.

3.4.24. Exercise. Convince yourself that the proofs of Theorem 3.4.4 and Corollary 3.4.5 work for polynomials over any integral domain (see Comment 3.4.22). In other words, if I is an integral domain and P a polynomial of degree $d \geq 0$ with coefficients in I, then P has at most d zeros in I.

3.4.25. Exercise. Corollary 3.4.8 states, with our new terminology, that working with rational polynomials modulo an irreducible polynomial yields an integral domain. This means that the (complicated looking) Theorem 3.4.11 actually follows immediately from Exercise 3.4.24!

Convince yourself that the proof of Theorem 3.4.7 also proves the following statement: let K be a field; we consider the arithmetic of polynomials whose coefficients belong to K. If H is irreducible (over K), then there are no zero divisors modulo H. That is, if $P \cdot Q \equiv 0 \pmod{H}$, then $P \equiv 0$ or $Q \equiv 0$ modulo H. (Even more is true: the polynomials over K form a field modulo H.)

3.5. Polynomials and modular arithmetic

In the preceding section, we studied arithmetic modulo a polynomial. But there is another possibility of combining polynomials and modular arithmetic! Consider e.g. $P := 2X^4 + X^3 - 3X^2 + 5$ and $Q := 7X^3 + X^2 - 4X - 1$. If we reduce the coefficients modulo 2, then P becomes $P \equiv 0X^4 + 1X^3 + 1X^2 + 1 = X^3 + X^2 + 1 \pmod{2}$ since $2 \equiv 0$, $1 \equiv 1$, $-3 \equiv 1$, and $5 \equiv 1$ modulo 2. In the same way, $Q \equiv X^3 + X^2 + 1$ modulo 2.

So the polynomials P and Q are *congruent modulo* 2. This is precisely the phenomenon that we will study now – we are no longer interested in the exact coefficients of a polynomial, but only their remainder after division by a natural number n.

3.5.1. Definition (Congruence of polynomials modulo n)**.**
Let $n \geq 2$ be a natural number and let P, Q be integer polynomials. Let d be the larger of the degrees of P and Q; we write $P = a_d X^d + \cdots + a_1 X + a_0$ and $Q = b_d X^d + \cdots + b_1 X + b_0$ with $a_0, \ldots, a_d, b_0, \ldots, b_d \in \mathbb{Z}$.

The polynomials P and Q are called **congruent modulo** n if $a_j \equiv b_j \pmod{n}$ for all $j \leq d$. In this case we write

$$P \equiv Q \pmod{n}.$$

We do not really need any new ideas to discuss this concept because the considerations from the last section can be carried over in a simple way. However, it might take a little while to get used to modular arithmetic with polynomials, so we shall work things out in little steps. After all, we will need these ideas later to understand the primality test of Agrawal, Kayal, and Saxena! In the following, we use the same concepts for polynomials modulo n that we introduced for integer polynomials in the previous section. For example the **degree modulo** n of a polynomial P is the largest number k such that the coefficient of X^k is not congruent to zero modulo n. We denote this degree by $\deg_n(P)$. So $\deg_2(2X^4+X^3-3X^2+5) = 3$. Similarly, a number $x \in \mathbb{Z}$ with the property that $P(x) \equiv 0 \pmod{n}$ is called a **zero of** P **modulo** n. For a prime p, the number of zeros modulo p is again bounded by the degree of the polynomial (Exercise 3.5.11), but this is false for composite numbers (Exercise 3.5.18).

Examples. X^3+6X-2 is congruent to $X^3-2X+14$ modulo 8 because $1 \equiv 1$, $6 \equiv -2$, and $-2 \equiv 14 \pmod{8}$. Furthermore, 2 is a zero of X^3+3X+1 modulo 3 because $2^3+3\cdot 2+1 = 15$ is divisible by 3.

Let us point out a difference between polynomials modulo n and polynomials over $\mathbb{Z}$ or $\mathbb{Q}$. By definition, two polynomials are congruent modulo n if and only if corresponding coefficients are congruent modulo n. This is *not* the same as saying that $P(x) \equiv Q(x) \pmod{n}$ for all integers x. For example, observe that $X^2 - X \not\equiv 0 \pmod{2}$, but $x^2 - x \equiv 0 \pmod{2}$ for all $x \in \mathbb{Z}$. (See also Exercise 3.5.7.)

Next, we shall discuss division with remainder for polynomials modulo n. When carrying out polynomial long division, we need to be able to divide by the leading coefficient of P. By Theorem 3.1.4, this is possible modulo n if and only if this coefficient is coprime to n (and, in particular, if the polynomial is monic).

3.5.2. Theorem (Polynomial long division modulo n)**.**
Let $n \geq 2$ be a natural number and let Q be a polynomial. Let P be a non-constant polynomial whose leading coefficient is coprime to n. Then there are polynomials T and R such that $\deg R < \deg P$ *and*

$$Q \equiv T \cdot P + R \pmod{n}.$$

Furthermore, the polynomials T and R are unique modulo n.

Proof. If P is monic, then the theorem follows immediately from the division theorem for integer polynomials (Theorem 3.4.2). If P is not monic, then let a be the leading coefficient of P. By hypothesis, we can divide both Q and P by a modulo n, obtaining polynomials $\tilde{Q}$ and $\tilde{P}$. Note that $\tilde{P}$ is a monic polynomial, so we can divide $\tilde{Q}$ by $\tilde{P}$

with remainder. Multiplying the resulting congruence by a on both sides, we obtain the desired result. (Alternatively, we can carry out a polynomial long division modulo n directly.) Uniqueness of T and R modulo n follows just as in Theorem 3.4.2. ■

To illustrate the theorem, let us try dividing $Q := X^4 + 5X + 4$ by $P := 4X + 1$ modulo 5. We proceed just as in the proof we have just given and begin by transforming P into a monic polynomial. The inverse of 4 modulo 5 is 4 itself, so we calculate

$$\begin{aligned}\tilde{Q} &= 4 \cdot Q = 4X^4 + 20X + 16 \equiv 4X^4 + 1 \pmod 5 \quad \text{and} \\ \tilde{P} &= 4 \cdot P = 16X + 4 \equiv X + 4 \pmod 5.\end{aligned}$$

Now we divide $4X^4 + 1$ by $X + 4$ with remainder:

$$\begin{aligned}4X^4 + 1 &= (X+4)(4X^3 - 16X^2 + 64X - 256) + 1025 \\ &\equiv (X+4)(4X^3 + 4X^2 + 4X + 4) \pmod 5.\end{aligned}$$

Finally, multiply both sides of this congruence by 4 again; we obtain

$$X^4 + 4 \equiv (4X+1) \cdot (4X^3 + 4X^2 + 4X + 4) \pmod 5.$$

In particular, $4X + 1$ is a **divisor** of $X^4 + 4X + 4$ **modulo** 5.

3.5.3. Definition (Divisors modulo n)**.**
Let P and Q be polynomials and let $n \geq 2$ be a natural number. Then P is called a **divisor** of Q **modulo** n if there is a polynomial T such that $Q \equiv T \cdot P \pmod n$.

We are now ready to introduce a concept that will be central for the AKS algorithm: modular arithmetic with respect to a number n and a polynomial H at the same time.

3.5.4. Definition (Congruence modulo n and H)**.**
Let $n \geq 2$ and let H be a non-constant polynomial whose leading coefficient is coprime to n. Then we write

$$P \equiv Q \pmod{n, H}$$

if P and Q have the same remainder when divided by H modulo n.

Example. Let $n := 3$, $H := X+1$, $P := X^2+3$, and $Q := 2X^3+X-2$. Dividing P and Q by H with remainder, we obtain $P = H\cdot(X-1)+4$ and $Q = H \cdot (2X^2 - 2X + 3) - 5$. The remainders -5 and 4 are congruent modulo 3, so

$$X^2 + 3 \equiv 2X^3 + X - 1 \pmod{3, X+1}.$$

To conclude, we generalize the concepts introduced at the end of the last section, namely irreducible polynomials and polynomial zeros. Here we restrict ourselves to calculations modulo a prime number, which we know to have particularly nice properties.

3.5.5. Definition (Irreducible polynomials modulo p)**.**
Let p be a prime number. Then a polynomial H with $\deg_p(H) > 0$ is called **irreducible modulo** p if the following is true:

If P and Q are polynomials with $H \equiv P \cdot Q \pmod{p}$, then $\deg_p(P) = 0$ or $\deg_p(Q) = 0$.

Example. The polynomial $H = X^2 + X + 1$ is irreducible modulo 2. Assume otherwise. Then there is a divisor P such that $\deg_2(P) = 1$, and hence either $P \equiv X + 1$ or $P \equiv X \pmod{2}$. But neither of these two polynomials is a divisor of H modulo 2 – this can be seen by using polynomial long division or by observing that H has no zeros modulo 2. So, by contradiction, H is indeed irreducible modulo 2. On the other hand, H is *not* irreducible modulo 3 because

$$H = X^2 + X + 1 \equiv X^2 + 4X + 4 = (X + 2)^2 \pmod{3}.$$

Let p be a prime number and let H be a polynomial whose leading coefficient is not divisible by p. If P and Q are polynomials with

$$P(Q) \equiv 0 \pmod{p, H},$$

then (similarly to Definition 3.4.10) we say that Q is a **polynomial zero** of P (modulo p and H).

The following statement, which corresponds to Theorem 3.4.11, will play an important role in the proof of the theorem of Agrawal, Kayal, and Saxena.

3.5.6. Theorem (Number of polynomial zeros mod p and H)**.**
Let p be a prime number and let H be a polynomial that is irreducible modulo p. Furthermore let P be a polynomial that has degree $d \geq 0$ modulo p. Then P has at most d polynomial zeros that are pairwise not congruent modulo p and H.

Proof. The proof works just as in Theorem 3.4.11. First we show a statement that corresponds to Theorem 3.4.7: if H divides the product $B \cdot C$ modulo p, then H also divides one of the polynomials B and C modulo p. Here the arguments from Theorem 3.4.7 are applicable almost word for word. (We recall that, by Theorem 3.5.2, we can divide with remainder modulo p by any polynomial that is not congruent to the zero polynomial.) For the main statement of Theorem 3.5.6 we study, as in Theorem 3.4.11, polynomials in Y whose coefficients are themselves polynomials in X. We see that (modulo p and H) we can divide out a "linear factor" $Y - Q$ for every polynomial zero Q and obtain, in the end, a representation of P as

$$P \equiv \tilde{P} \cdot (Y - Q_1) \cdot (Y - Q_2) \cdots (Y - Q_k) \pmod{p, H},$$

where $\tilde{P}$ has no polynomial zeros. Now the theorem follows from the claim at the beginning of the proof. (See also Comment 3.5.19.) ■

Exercises.

3.5.7. Exercise. Let $n \geq 2$ be a natural number. Show that there is a polynomial P with $P \not\equiv 0 \pmod{n}$ but $P(x) \equiv 0 \pmod{n}$ for all $x \in \mathbb{Z}$.

3.5.8. Exercise.

(a) Prove that $X^4 + X^2 - 2 \equiv 0 \pmod{3, X^2 + X + 1}$.

(b) Prove that $2X^5 + 3X^3 + X^2 + 1 \equiv 5X \pmod{6, X^2 + 1}$.

(c) Find a polynomial P of degree at most two such that

$$2X^5 + 4X^2 + X + 5 \equiv P \pmod{7, X^3 + 2X^2 + 5X + 6}.$$

3.5.9. Exercise (!). Show that polynomial long division modulo a natural number n is efficient. (I.e. the running time is polynomial in $\log n$ and in the degree of the polynomials under consideration.) Deduce: if $n \geq 2$ and H is a monic polynomial, then there are efficient algorithms to compute sums and products of polynomials modulo n and H, as well as for the computation of powers of polynomials modulo n and H.

3.5.10. Exercise (P). Implement the algorithms in Exercise 3.5.9. (This requires the definition of a suitable data type for polynomials.)

3.5.11. Exercise (!). Let $n \geq 2$ be a natural number, let $a \in \mathbb{Z}$, and let P be a polynomial. Prove the following:

(a) a is a zero of P modulo n if and only if $(X - a)$ is a divisor of P modulo n.

(b) If $P \not\equiv 0 \pmod{n}$, then P has a decomposition

$$P \equiv (X - a_1) \cdots (X - a_m) \cdot Q \pmod{n}. \tag{3.5.12}$$

Here $m \geq 0$, the numbers $a_1, \ldots, a_m$ range from 0 to $n - 1$, and Q is a polynomial that has no zeros modulo n.

(c) If n is prime, then the numbers $a_1, \ldots, a_m$ are unique up to reordering. I.e. if

$$P \equiv (X - b_1) \cdots (X - b_k) \cdot R \pmod{n}$$

is another such decomposition, with $b_1, \ldots, b_k \in \{0, ..., n - 1\}$, then $m = k$ and the b_j agree with the a_j up to their ordering.

(d) If n is prime and $P \not\equiv 0 \pmod{n}$, then P has at most $\deg_n(P)$ zeros modulo n (up to congruence).

3.5.13. Exercise.

(a) Is $X^2 + 1$ irreducible modulo 2?

(b) Is $5X^2 + X + 1$ irreducible over $\mathbb{Q}$? Modulo 7?

3.5.14. Exercise. Show that for every natural number n there is a polynomial that is irreducible over $\mathbb{Q}$, but not irreducible modulo n.

3.5.15. Exercise (!). Let p be a prime and let P be a polynomial with $P \not\equiv 0 \pmod{p}$. Show:

(a) If $\deg_p(P) > 0$, then there is a monic polynomial H that is irreducible modulo p and divides P modulo p.

(b) There are $m \geq 0$, $a \in \mathbb{Z}$, and monic irreducible polynomials $H_1, \ldots, H_m$ such that

$$P \equiv a \cdot H_1 \cdots H_m \pmod{p}.$$

(In other words, P has a decomposition into irreducible factors modulo p.)

3.5.16. Exercise. Show that the polynomials $H_1, \ldots, H_m$ demonstrated to exist in Exercise 3.5.15 are unique modulo p up to their order. (Use the statement at the beginning of Theorem 3.5.6, which is an analog to Theorem 3.4.7.)

Further Exercises and Comments.

3.5.17. Let H be a monic polynomial. If H is irreducible modulo p, then H is also irreducible over $\mathbb{Z}$, and hence (by Exercise 3.4.21) over $\mathbb{Q}$. (The converse is false; see Exercise 3.5.13.)

3.5.18. Exercise. If p is prime, it follows from Exercise 3.5.11 or from Exercise 3.4.24 that a polynomial of degree $d \geq 0$ has at most d zeros modulo p no two of which are congruent modulo p. In contrast, find a composite natural number n and a polynomial for which the number of zeros modulo n is larger than the degree.

3.5.19. Theorem 3.5.6 illustrates the power of the abstract point of view that we developed in the further comments at the end of the preceding section. Indeed, this theorem, whose statement looks quite complicated, follows immediately from Exercises 3.4.25 and 3.4.24. The reason is that the numbers modulo p form a field, and hence polynomials modulo p and H form an integral domain by Exercise 3.4.25.

By Exercise 3.4.24, polynomials whose coefficients belong to an integral domain have at most as many zeros as indicated by their degree. Hence we have proved the statement of Theorem 3.5.6!

3.5.20. Exercise. Just as it is not always simple to recognize a number as a prime, it is not so easy to tell whether a polynomial is irreducible (over $\mathbb{Q}$). In this exercise we study a condition that is often useful, the **Eisenstein irreducibility criterion**:

Let P be a polynomial of degree d and let p be a prime such that the leading coefficient a_d of P is coprime to p, all other coefficients are divisible by p, but such that the constant coefficient of P is not divisible by p^2. Then P is irreducible over $\mathbb{Q}$.

For example, $X^4 + 2X^3 + 24X + 6$ satisfies these conditions for $p = 2$. Prove this irreducibility criterion along the following steps:

(a) Suppose, by contradiction, that we can write $P = Q \cdot R$, where Q and R are integer polynomials neither of which is equal to ± 1. Then $Q \cdot R = P \equiv a_d X^d \pmod{p}$.

(b) Show first that Q and R must both be nonconstant.

(c) Conclude that $Q \equiv bX^{d_1}$ and $R \equiv cX^{d_2} \pmod p$ for suitable numbers b, c coprime to p and $d_1, d_2 > 0$.

(d) In particular, the constant coefficients of Q and R must be divisible by p. Derive a contradiction from this fact.

(e) So we have proved that P is irreducible over $\mathbb{Z}$. By Exercise 3.4.21, P is also irreducible over $\mathbb{Q}$.

Further reading

The book *Numbers, Groups and Codes* [**HP**] contains an introduction to modular arithmetic, including the theorems of Fermat, Euler, and Lagrange. For a more extensive treatment of number theory, we recommend a real classic: *An Introduction to the Theory of Numbers* by Hardy and Wright [**HW**]. Readers who are interested in the area of algebraic number theory will find the book *Problems in Algebraic Number Theory* [**ME**] to be a wonderful introduction with many exercises. For further reading about the ideas of abstract algebra which we have hinted at in some of the further comments and exercises, we recommend for example *A First Course in Abstract Algebra* by Fraleigh [**Frl**], which, in particular, covers the arithmetic of polynomials.

Chapter 4

Prime numbers and cryptography

We begin this chapter with an overview of the history of cryptography and of its basic concepts. Then we discuss the most important example of public-key cryptography: the RSA method. Most of the remainder of the chapter is concerned with the distribution of prime numbers; in particular, we give an elementary proof of a weak version of the celebrated **prime number theorem**. We conclude the first part of the book by finally presenting an efficient (but randomized) method of testing for primality: the **Miller-Rabin algorithm**.

4.1. Cryptography

Cryptography refers to the development, study, and improvement of encryption systems – its goal is to find methods for encrypting messages that are as secure and as easy to handle as possible. Its counterpart is **cryptanalysis**, which tries to use encrypted messages *without* knowledge of encryption keys to find the method of encryption and decipher the messages. Thus the history of cryptography, which goes back more than 2000 years, can be seen as a competition between cryptographers and cryptanalysts: new, more complicated, and more secure codes are developed, while the methods used to "crack" them become ever more sophisticated. Already around

1900 BC, an Egyptian inscription in a pharaoh's grave used unusual hieroglyphics whose meaning is still unknown today – possibly an early example of written cryptography. In the early days of secret communication, messages would often simply be hidden, for example by scratching them into boards that were then covered in wax. If encryption was used at all, it was in the simplest possible manner: by changing the order of letters. In this way, the original message (called the **plaintext**) is transformed into the **ciphertext**, i.e. the encrypted message. Frequently the **skytale** is mentioned as the first known instrument for more elaborate encryption. It is thought that this wooden rod was used to roll up a thin band of papyrus on which the message would then be written. After unrolling it, the result would be an unreadable mess of letters, which could be deciphered by using another wooden rod of the same diameter. For example, the Spartan general Pausanias is supposed to have used a skytale in this manner around 475 BC. However, historians disagree about whether the skytale was really used for encryption or not; see [**Ke**].

The transmission of relatively simply encrypted and additionally hidden messages seems to have been the safest known method of secret communication for quite some time. Only around 170 BC was a new method developed: the **Polybius square**, which involved transforming letters into numerical symbols.

A very well-known method of encryption is supposed to have been used by Julius Caesar to send secret messages to his troops and has become known as the **Caesar cipher**. This is again a **mono-alphabetic cipher**, which means that the letters of the alphabet are scrambled, and this **ciphertext alphabet** is then used to encrypt the entire text. The Caesar cipher simply shifts all letters of the alphabet three places to the right, which is a rather insecure method in the long run. Once the principle is understood, it does not really help to shift by a different number of places (which Caesar is thought to have done later) – the code is easy to decipher.

Around 1000 AD, Arabic cryptanalysts developed a systematic method for breaking mono-alphabetic encryption: **frequency analysis**. The idea is as follows: if we know in which language the plaintext was written, then we also know which letters or letter combinations are common (e.g., in English, “e” and “t”, or “th” and “he”) and can try to find these in the ciphertext. The more encrypted messages we have intercepted, the better this methods works since the probability rises that letters will really be distributed similarly as in the complete language. Once a few letters have been identified, the rest can quickly be deduced from the context. The discovery of this method exposed the limits of mono-alphabetic ciphers, and around 1500 AD the first **poly-alphabetic ciphers** were developed. Here not just one but several different ciphertext alphabets are used, which are repeatedly switched between during encryption. This was a milestone in the history of cryptography, as the characteristics of the language could no longer be found in the ciphertext and frequency analysis was rendered useless. The idea was first formulated by the mathematician Leon Battista Alberti in the fifteenth century, but it was the French diplomat Blaise de Vigenère who made the method famous. He used twenty-six different ciphertext alphabets – shifted by one, two, three, four, ... letters to the right.

For a long time this was a very secure, almost unbreakable encryption method, but it had a weak point: the keyword, which was used to determined which of the twenty-six alphabets are used in which order. Once tests were developed to find the length of the keyword (such as the Kasiski test, named for the Prussian officer Friedrich W. Kasiski, who discovered and exploited this weakness in 1863), frequency analysis could be applied to groups of letters in the ciphertext.

Digraphic substitution ciphers also attempt to render frequency analysis ineffective, by always treating two letters together while obeying certain rules. An example is the Playfair cipher, developed by the English physicist Charles Wheatstone (1802–1875).

A much more complex poly-alphabetic cipher was realized by the **ENIGMA**, an encryption machine used by Germany during World War II that automatically alternated between over 100 000 different alphabets. It was cracked in 1940 by a team headed by the English mathematician Alan Turing, who was able to narrow down the collection of possible keys and designed a machine that could test the remaining possibilities within a short time span.

All crypto-systems that we have described so far have the disadvantage of being **symmetric**, i.e. the same key is used for sending and receiving messages. So the encryption can be cracked once the encryption algorithm and the key are known. This means that the security of the message depends heavily on the channel that is used to transmit the key, which makes this an ideal point of attack for code breakers. The notion of **public-key cryptography**, developed in 1975, removes this weakness – another milestone.

The idea is to use two different keys: a “public key”, which is used for encryption, and a “private key”, which allows the decryption of an encrypted message. Both keys are generated by the recipient, who publishes the public key (for example on the Internet) but keeps the private key secret. Now anyone in the world can encrypt a message using the public key, but only the owner of the private key will be able to decrypt it again. This completely circumvents the problem of key transmission, which had remained unsolved for centuries! The first, most well-known, and most commonly used procedure of this kind is the **RSA method**, named for its inventors Ronald Rivest, Adi Shamir, and Leonard Adleman, which will be explained thoroughly in the next section.

For further reading on the history of cryptography, we recommend *The Code Book* by Simon Singh [**S**].

4.2. RSA

At first glance, the principle of public-key cryptography appears daring: how could it be that we can carry out a calculation but then be unable to reverse it without additional information?

But we have already encountered mathematical operations that can be carried out easily, but not reversed efficiently. (These are often called “one-way functions”.) For example, it is easy to multiply two large prime numbers together, but to recover the two prime factors from this product is much harder. Exactly this – the assumption that the factorization problem cannot be solved efficiently – is the basis of the RSA system. We can outline the underlying idea as follows.

Let p and q be two large prime numbers. (Here we really mean "large"; in practice they will have several hundred digits or more.) We consider their product $n := p \cdot q$. There are a number of operations modulo n that are easy to carry out if we know p and q but otherwise are very difficult. To see this, we recall the Fermat-Euler Theorem, which states that $a^{\varphi(n)} \equiv a \pmod{n}$ provided that $a \in \mathbb{Z}$ and $n \geq 2$ are coprime. If we know p and q, then we can compute $\varphi(n)$ easily: by Exercise 3.2.17, we have

$$\varphi(n) = \varphi(p) \cdot \varphi(q) = (p-1) \cdot (q-1).$$

On the other hand, it is impossible to find the value of $\varphi(n)$ without also finding p and q (which we believe to be an intractable problem); see Exercise 4.2.2. If we now pick a number $e \in \{1, \ldots, \varphi(n)\}$ that is coprime to $\varphi(n)$, then:

(a) The power $R(M) := M^e \bmod n$ can be computed easily from e and n, for every number M that is coprime to n, by using the power algorithm. (See Exercise 3.1.14.)

(b) To recover the original element M from the power $R(M)$ using only n and e, on the other hand, seems to be much more difficult. (See Comment 4.2.4.)

With knowledge of $\varphi(n)$, however, it becomes easy: using the Euclidean algorithm, we compute an *inverse* d of e modulo $\varphi(n)$, i.e. a number d with $d \cdot e = 1 + k \cdot \varphi(n)$ for some integer k. By the Fermat-Euler Theorem, we see that

$$\begin{aligned} R(M)^d &\equiv (M^e)^d = M^{e \cdot d} = M^{1+k\cdot\varphi(n)} \\ &= (M^{\varphi(n)})^k \cdot M \equiv 1^k \cdot M = M \pmod{n}. \end{aligned}$$

This means that if we know d and n, we can recover M from $R(M)$ simply by computing $(R(M))^d \bmod n$.

So exponentiation modulo n is exactly the kind of operation that we were looking for: it is easy to compute but can be reversed only using additional information. The public key consists of the numbers n and e. Using these, the number $R(M)$, which is our ciphertext, can be calculated easily. The private key consists of the numbers n and d; they allow us to recover M from the encrypted message $R(M)$.

More precisely, we use the following algorithm to generate keys:

```
Algorithm RSA-KEY
  1. Randomly generate two large prime numbers $p$ and $q$. Set $n := p \cdot q$.
  2. Compute $\varphi(n) = (p-1) \cdot (q-1)$.
  3. Randomly choose a number $e \in \{1, \ldots, \varphi(n)-1\}$ that is coprime to $\varphi(n)$.
  4. Use the Euclidean algorithm to find the number $d \in \{1, \ldots, \varphi(n)-1\}$ with $e \cdot d \equiv 1 \pmod{\varphi(n)}$.
  5. The public key consists of the numbers $n$ and $e$.
  6. The private key consists of the numbers $n$ and $d$.
```

Using the public and private keys that are generated in this manner, en- and decryption are easy to carry out:

```
Algorithm RSA-ENCRYPTION
Input: A number $M$ coprime to $n$.
  1. Compute $V := M^e \bmod n$.
  2. $V$ is the ciphertext.
```

```
Algorithm RSA-DECRYPTION
Input: A ciphertext $V$.
  1. Compute $M := V^d \bmod n$.
  2. $M$ is the plaintext.
```

To be able to implement the algorithm RSA-KEY, we still need to clarify how we can randomly pick two large prime numbers in Step **1**. In the next section, we will see that this depends on being able to efficiently test a number for primality. As already mentioned in the introduction, this means that RSA encryption relies on the following principle:

> It is easy to test whether a number n has a non-trivial divisor, but it is hard to find such a divisor!

Exercises.

4.2.1. Exercise. We will carry out a sample RSA encryption by hand. Of course this is possible only with very small numbers.

(a) Carry out the algorithm RSA-KEY for $p = 13$, $q = 17$, and $e = 5$ to obtain private and public keys.

(b) Encrypt the number $M = 10$ using RSA-ENCRYPTION.

(c) Use the algorithm RSA-DECRYPTION to decrypt the ciphertext obtained in (b). (This is possible by hand, but rather cumbersome, so it is advisable to use a calculator.)

4.2.2. Exercise. Let $n = p \cdot q$ be the product of two prime numbers. Show that the primes p and q are exactly the two solutions of the quadratic equation

$$x^2 + x(\varphi(n) - n - 1) + n = 0.$$

(*Hint:* Begin with the equation $\varphi(n) = (p - 1) \cdot (q - 1)$ and apply the binomial formula. Then use the fact that $q = n/p$ to obtain a quadratic equation for p.)

It follows that it is possible to determine the factors p and q once both n and $\varphi(n)$ are known. In other words, in this case, computing $\varphi(n)$ from n is just as difficult as factorizing n.

Further Exercises and Comments.

4.2.3. Whitfield Diffie and Martin Hellman developed the principle of public-key cryptography at Stanford University in California and described it in a research article from 1976. In the following year, Rivest, Shamir, and Adleman designed the RSA method at the Massachusetts Institute of Technology (MIT) in Boston.

4.2.4. The design of the RSA algorithm suggests three obvious ways to try to systematically break the encryption:

(a) Find the factors p and q by factorizing n.

(b) Compute the number $\varphi(n)$ somehow (without first factorizing n).

(c) Somehow calculate the secret key d directly from the numbers n and e.

In Exercise 4.2.2, we saw that computing $\varphi(n)$ would also allow factorization of n. Similarly, it is known that determining the secret key d would also allow us to find the prime factors of n. Hence, if there really is no efficient method of factorizing n, none of the above methods can be successful. (Compare [**RSA**, Section IX] or [**CM**].)

4.2.5. To ensure secure communication using the RSA method in practice, it is necessary to make some further considerations. For example, one should make sure that the decryption exponent d is not too small when compared to n, and the plaintext should be encoded in a suitable way *before* encryption in order to prevent certain cryptanalytic attacks.

For a short overview of such issues, we refer the reader to the article [**Rob**], which was published in the *SIAM News* to celebrate the *Alan Turing Award* for Rivest, Shamir, and Adleman in 2003.

4.2.6. Although factorizing a composite number n is considered to be an intractable problem, there are nonetheless methods that are considerably better than simply looking for possible factors of n. For example, such algorithms (and a lot of computer time!) were used in 1991 to factorize the 100-digit number "RSA-100", which would be impossible with elementary methods (recall Comment 1.5.3). The largest such number that has been factorized to date (in 2005) has 200 digits. The necessary computations were carried out in parallel on many computers all over the world (and would have taken around 75 years if only a single machine had been used). Numbers with "only" up to about 40 to 50 digits, such as the number 1 726 374 899 084 624 209, can be factorized in seconds by one of today's standard home computers. The reader is invited to try this, for example on the website `http://www.alpertron.com.ar/ECM.HTM`.

4.3. Distribution of primes

So far, we have ignored the question of how exactly the RSA algorithm can randomly pick two large prime numbers. Without such a method, it would not even be possible to create the public and private keys.

A first idea might be to pick a sufficiently large number k at random and then compute the k-th prime number. Since we do not know any practicable method for computing the k-th prime, this unfortunately would not work. Instead, we use the following method to randomly pick a prime that is less than or equal to some number n:

```
ALGORITHM RANDOM-PRIME
Input:  A number $n \geq 2$.
  1. Randomly choose a number $k \leq n$.
  2. Test whether $k$ is prime.
  3. If yes, output $k$. Otherwise, return to Step 1.
```

Finding methods that can be used to *efficiently* test for primality in Step **2** is the main goal of our book. An answer that is absolutely sufficient for practical purposes will be given in Section 4.5. For now, we instead devote ourselves to considering how many different numbers k we need to test, on average, until we find a prime number. To answer this, we need to consider *how many primes* $p \leq n$ *there are at all.* Therefore we define

$$\pi(n) := \#\{p \leq n : p \text{ is prime}\}$$

and note:

4.3.1. Lemma.
The average number of repetitions required in algorithm RANDOM-PRIME is

$$\frac{n}{\pi(n)}.$$

Proof. In every repetition, the probability w of picking a prime in Step **1** is exactly

$$w = \frac{\pi(n)}{n}.$$

Thus our question can be rephrased as follows: we repeatedly throw a coin with the property that the probability of throwing "heads" is w and the probability of throwing "tails" is $1-w$. How often do we have to throw the coin, on average, until we obtain "heads" for the first time? Exercise 2.5.6 answers this question; as claimed, the expected number of repetitions is

$$\frac{1}{w} = \frac{n}{\pi(n)}. \qquad \blacksquare$$

So the situation is as follows: if the number of primes up to n is so large that $n/\pi(n)$ grows at most polynomially in $\log n$ (the size of our input), everything is fine. Otherwise we really have a problem since we cannot efficiently generate keys to be used in RSA! The celebrated **prime number theorem**, which is over a hundred years old, ensures that the latter does not happen.

4.3.2. Theorem (Prime number theorem). *$\pi(n)$ asymptotically behaves like the function $n/\ln(n)$. More precisely, if $\varepsilon > 0$ is any (arbitrarily small) real number, then*

$$(1-\varepsilon)\cdot\frac{n}{\ln(n)} \leq \pi(n) \leq (1+\varepsilon)\cdot\frac{n}{\ln(n)}$$

for all sufficiently large natural numbers n.

Thus, if we are able to efficiently test numbers for primality, we can use algorithm RANDOM-PRIME to efficiently generate large prime numbers and use these for the RSA system.

A proof of the prime number theorem is far beyond the scope of our book, but we shall not require the full strength of the theorem for our purposes. Therefore we will be content with the following version, which is proved by elementary means in the next section.

4.3.3. Theorem (Weak version of the prime number theorem). *There is a real number $C > 0$ such that*

$$\pi(n) \geq C\cdot\frac{n}{\log n}$$

for every natural number $n \geq 2$.

Finally, we would like to remark that we can use these results also to answer another remaining question, namely how to efficiently choose the number e in Step **3** of algorithm RSA-KEY. The details are treated in Exercise 4.3.4.

Exercises.

4.3.4. Exercise. Let $n \geq 2$ be a natural number.

(a) Show that n has at most $\log(n)$ different prime divisors.

(b) Deduce, using Theorem 4.3.3, that there is a constant $C' > 0$, independent of n, such that

$$\varphi(n) \geq C'\cdot\frac{n}{\log n}.$$

(c) Show that it is possible to efficiently and randomly choose a number $e < n$ that is coprime to n. Conclude that Step **3** of RSA-KEY can be carried out efficiently.

Further Exercises and Comments.

4.3.5 (History of the prime number theorem)**.** The prime number theorem was conjectured, but not proved, by Carl Friedrich Gauss at the age of 15 in 1792. The number theorist Adrien-Marie Legendre independently formulated the same conjecture. Only in 1851 was Chebyshev able to prove the weak version of the prime number theorem (Theorem 4.3.3), together with a similar upper bound for $\pi(n)$ (see [**De**, Chapter 10]). It took almost another half century until Hadamard and de la Vallée Poussin, independently of each other, found the first complete proof of the prime number theorem in 1896. A quite short modern proof (which, however, requires some background from *complex analysis*) can be found in the article [**Za**]. The book [**J**], which does not require much prior knowledge, provides an extensive discussion of the prime number theorem and its proofs.

4.3.6 (The Riemann Hypothesis)**.** While the prime number theorem gives us some rough information on how many primes numbers there are, there are still many open questions about the distribution of primes! Indeed, already Gauss conjectured that the function

$$\mathrm{Li}(n) := \int_2^n \frac{\mathrm{d}t}{\ln t},$$

called the **integral logarithm**, approximates the prime counting function $\pi(n)$ even better than the function $n/\ln n$.

The question is: how good is the approximation of $\pi(n)$ by $\mathrm{Li}(n)$? In other words, how large is the error $|\pi(n) - \mathrm{Li}(n)|$? In 1901, von Koch [**vK**] proved that the **Riemann Hypothesis** is true if and only if this error term can be estimated as

$$|\pi(n) - \mathrm{Li}(n)| = O(\sqrt{n}\ln n).$$

This is one of the reasons that the Riemann Hypothesis (whose original statement can be found in Appendix A) is one of the most famous unsolved problems in mathematics and the subject of ongoing research.

4.4. Proof of the weak prime number theorem

In this section we prove Theorem 4.3.3. The methods used, while interesting, are not relevant for the remainder of the book. Readers who are eager to learn about the AKS primality test are therefore encouraged to skip this section and come back to it later. Our proof will exploit a close relationship between the prime counting function

$\pi(n)$ and the least common multiple

$$v(n) := \operatorname{lcm}(1, 2, 3, \ldots, n)$$

of the numbers $1, 2, 3, \ldots, n$.

What exactly is this relationship? Well, every prime number $p \leq n$ must occur in the prime decomposition of $v(n)$. So if $\pi(n)$ is large, then $v(n)$ must become large also. Conversely, the number $v(n)$ can *only* become large if there are enough prime numbers that can appear in its decomposition. The following lemma contains a more precise statement.

4.4.1. Lemma (Upper and lower bounds for $v(n)$)**.**
$\sqrt{n}^{\pi(n)} \leq v(n) \leq n^{\pi(n)}$ *for all* $n \in \mathbb{N}$.

Proof. First, we decompose $v(n)$ into its prime factors. Let $k \in \mathbb{N}_0$, let $e_1, ..., e_k \in \mathbb{N}$, and let $p_1, ..., p_k$ be prime numbers such that

$$v(n) = p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k}.$$

As remarked above, the prime factors of $v(n)$ are exactly the primes p with $p \leq n$. So we have $k = \pi(n)$.

Furthermore, for all $1 \leq j \leq k$, the definition of $v(n)$ means that $p_j^{e_j}$ is the highest power of p_j which divides some number $m \leq n$. In particular, we have $p_j^{e_j} \leq n$ for all n. On the other hand, we also see that $p_j^{e_j} \geq \sqrt{n}$ since $p_j^{e_j+1} > n$. Overall,

$$v(n) = p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k} \leq n \cdot n \cdots n = n^k = n^{\pi(n)} \qquad \text{and}$$
$$v(n) = p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k} \geq \sqrt{n} \cdot \sqrt{n} \cdots \sqrt{n} = \sqrt{n}^{\pi(n)}. \qquad \blacksquare$$

The preceding lemma implies that, instead of proving the weak prime number theorem, we can just as well establish a corresponding statement for $v(n)$.

4.4.2. Corollary (Reformulation of the weak prime number theorem)**.**
The weak prime number theorem holds if and only if there is a constant K such that $v(n) \geq 2^{Kn}$ for all $n \geq 2$.

Proof. Let $n \in \mathbb{N}$ with $n \geq 2$. If the weak prime number theorem holds, we have $\pi(n) \geq Cn/\log n$ for some $C > 0$ independent of n. By Lemma 4.4.1 this means that we also have

$$v(n) \geq \sqrt{n}^{\pi(n)} = \left(2^{\frac{\log n}{2}}\right)^{\pi(n)} = 2^{\frac{\pi(n)\log n}{2}} \geq 2^{\frac{Cn}{2}},$$

so we set $K := C/2$ and are done.

Conversely, assume there is a constant K with $v(n) \geq 2^{Kn}$ for all n. By Lemma 4.4.1, we have $\pi(n) \geq \frac{\log v(n)}{\log n}$ and therefore

$$\pi(n) \geq \frac{\log v(n)}{\log n} \geq K\frac{n}{\log n}. \qquad \blacksquare$$

So all that remains to be done is to bound $v(n)$ from below. To do so we will – for the only time in this book – use a method from calculus to obtain a result on natural numbers (see Comment 4.4.6).

4.4.3. Theorem (Lower bound for $v(n)$)**.**
$v(n) \geq 2^{n-2}$ for all natural numbers n.

Proof. Since we always have $v(n) \geq 1$, the claim is true for $n = 1$ and $n = 2$. So we may assume that $n > 2$.

The proof is elementary, but rather clever, so we begin by explaining the underlying idea. Let us assume that we can find some integers $a_1, a_2, \ldots, a_n$ in such a way that the sum

$$s := \frac{a_1}{1} + \frac{a_2}{2} + \frac{a_3}{3} + \cdots + \frac{a_n}{n} \tag{4.4.4}$$

is very small, but positive. Since $v(n)$ is a common multiple of all numbers from 1 to n, the number $s \cdot v(n)$ would then be a positive integer. In particular we have $s \cdot v(n) \geq 1$, and division by s yields a lower bound for $v(n)$.

How can we make the sum s very small? Here calculus provides a handy trick because we can write s as the integral of a polynomial with integer coefficients. After all,

$$\frac{a_k}{k} = \int_0^1 a_k X^{k-1} dX.$$

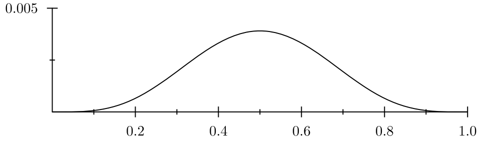


Figure 4.1. The graph of the function $P(x) = x^m(1-x)^m$ for $m = 4$.

So, if $P = a_1 + a_2X + a_3X^2 + \cdots + a_nX^{n-1}$ is some arbitrary polynomial of degree at most $n-1$ and with integer coefficients, then we can view P as a polynomial function of real numbers and define $I(P) := \int_0^1 P(x)dx$. This is a sum just of the form given in (4.4.4). If $I(P) > 0$, then we see, as above,

$$v(n) \geq \frac{1}{I(P)}.$$

Our goal now is to find a polynomial P such that the integral $I(P)$ is as small as possible. Suppose that n is odd, i.e. $n = 2m+1$. Then we look at

$$P := X^m(1-X)^m.$$

The graph of P as a function (Figure 4.1) looks like a "squashed" upside-down parabola. Its maximum is taken at $x = 1/2$. At this point, we have

$$P(1/2) = \frac{1}{2^{2m}}.$$

In particular, $I(P) \leq 1/2^{2m}$, and we have shown that $v(n) \geq 2^{n-1}$ for odd n. If n is even, then this implies that $v(n) \geq v(n-1) \geq 2^{n-2}$, as desired. ■

To conclude the proof of the weak prime number theorem, we just note that $n-2 \geq n/2$ for $n \geq 4$, and thus

$$v(n) \geq 2^{n-2} \geq 2^{n/2}. \tag{4.4.5}$$

Since $v(2) = 2$ and $v(3) = 6$, the inequality (4.4.5) also holds for $n = 2$ and $n = 3$. By Corollary 4.4.2, we have proved the weak prime number theorem (with $C = 1/2$). ■

Further Exercises and Comments.

4.4.6. Integral calculus, together with **differential calculus**, is one of the cornerstones of **mathematical analysis**. For readers who are not familiar with its basic principles, we give a short overview for completeness and refer e.g. to [**Br**] for further reading.

If f is a continuous function of real numbers (say, a polynomial) and $a < b$, then the **integral**

$$\int_a^b f(x)dx$$

denotes the *area* enclosed by the graph of the function f and the x-axis. (Here regions where the function takes negative values are counted as *negative area*.) Alternatively, we could say that

$$\frac{\int_a^b f(x)dx}{b-a}$$

is the *average value* of the function f on the interval $[a, b]$. Our proof of Theorem 4.4.3 only uses the following simple rules for working with integrals:

(a) The integral of the sum of two functions is exactly the sum of the integrals of these functions. (This should be plausible from the interpretation of the integral as an average.)

(b) If M is an upper bound for the function f, i.e. $f(x) \leq M$ for all $x \in [a, b]$, then

$$\int_a^b f(x)dx \leq M \cdot (b-a).$$

Again, this should be clear; if the function is never larger than M, then the same is true of its average value.

(c) If $a \in \mathbb{R}$ and $k \in \mathbb{N}_0$, then $\displaystyle\int_0^1 ax^k dx = \frac{a}{k+1}$.

4.4.7. The idea of the proof of Theorem 4.3.3 is taken from [**N**].

4.5. Randomized primality tests

In order to be able to use the RSA encryption method, we still need a way to test a number for primality efficiently and reliably. So far we do not know any such algorithms; our best attempt was the Fermat test in Section 3.3. This randomized test is efficient and often gives us the right answer. However, due to the existence of Carmichael numbers, it is not always reliable.

Before we treat the deterministic primality test of Agrawal, Kayal, and Saxena in the second part of the book, we now present a method that completely solves the problem for all practical purposes. This is the "Miller-Rabin primality test", an efficient Monte Carlo algorithm for the problem COMPOSITES. It is an extension of the Fermat test based on the following property of prime numbers:

4.5.1. Lemma (Roots of unity modulo a prime)**.**
Let p be prime and let $x \in \mathbb{Z}$ be such that $x^2 \equiv 1 \pmod{p}$. Then $x \equiv 1 \pmod{p}$ or $x \equiv -1 \pmod{p}$.

Proof. The hypothesis implies that $x^2 \equiv 1 \pmod{p}$, and thus

$$(x+1)(x-1) = x^2 - 1 \equiv 0 \pmod{p}.$$

By Corollary 1.3.5, we thus have $x + 1 \equiv 0$ or $x - 1 \equiv 0$. In other words, $x \equiv -1$ or $x \equiv 1$ modulo p. ■

What does this have to do with Fermat's test? For a (randomly chosen) number $a \in \{1, \ldots, p-1\}$, this test checks whether

$$a^{p-1} \equiv 1 \pmod{p} \tag{4.5.2}$$

holds. If p is prime, then this is always true. Since all primes $p \neq 2$ are odd, the number $p - 1$ is usually even. Hence we can rewrite (4.5.2) as

$$\left(a^{(p-1)/2}\right)^2 \equiv 1 \pmod{p}.$$

By Lemma 4.5.1, $a^{(p-1)/2}$ must be congruent to 1 or -1 modulo p.

We develop this idea a little further and write $p-1 = d \cdot 2^\ell$, where d is odd. For example, for $p = 13$ we would have $p - 1 = 3 \cdot 2^2$, i.e. $d = 3$ and $\ell = 2$. Then

$$a^{p-1} = a^{d \cdot 2^\ell} = \left(a^d\right)^{2^\ell} = \left(\ldots\left(\left(a^d\right)^2\right)\ldots\right)^2.$$

We see that either the initial number a^d is congruent to 1 or one of the numbers that occur during the repeated squaring has to be congruent to -1 modulo p. So we have found a new property of prime numbers that improves on Fermat's Little Theorem.

4.5.3. Theorem (Theorem of Fermat-Miller)**.**
Let p be an odd prime number. Write $p - 1 = d \cdot 2^{\ell}$, where $d, \ell \in \mathbb{N}$ and d is odd. If $a \in \mathbb{Z}$ is not a multiple of p, then

(a) either $a^d \equiv 1 \pmod{p}$

(b) or $a^{2^i \cdot d} \equiv -1 \pmod{p}$ *for some i in the range from* 0 *to* $\ell - 1$.

Proof. For brevity, let us set $b := a^d$. By Fermat's Little Theorem, we know that $b^{2^{\ell}} \equiv 1 \pmod{p}$. If $b \equiv 1$, then the first case of the theorem holds and we are done. Otherwise, let i be the *first* index between 0 and $\ell - 1$ such that $b^{2^{i+1}} \equiv 1$.

By Lemma 4.5.1, either $b^{2^i} \equiv 1$ or $b^{2^i} \equiv -1$. The first case is impossible by our choice of i as the *first* index with the stated property, and it follows that $b^{2^i} \equiv -1$ as claimed. ■

In Exercises 4.5.6 and 4.5.7, the reader is invited to convince herself that the theorem of Fermat-Miller really is an improvement over Fermat's Theorem when it comes to detecting composites. In particular, we can recognize at least some of the awkward Carmichael numbers, which cause problems in the Fermat test, as composites. This is reason enough to formulate a new primality test!

Algorithm MILLER-RABIN

Input: A natural number $n > 1$.

1. If n is even and $n > 2$, or if n is a power of some other natural number, answer "n is composite".
2. Otherwise write $n-1 = d{\cdot}2^{\ell}$ and choose a number a from 1 to $n - 1$ at random.
3. If $\gcd(a, n) > 1$, answer "n is composite".
4. Otherwise compute $b := a^d \bmod n$.
5. If $b = 1$, then answer "n is probably prime".
6. Otherwise compute $b, b^2, b^4, \ldots, b^{2^{\ell-1}} \pmod{n}$.
7. If none of these numbers are congruent to -1 modulo n, then answer "n is composite".
8. Otherwise answer "n is probably prime".

(Here we excluded the even numbers and perfect powers at the beginning only for convenience; see Comment 4.5.9.)

By Theorem 4.5.3, this algorithm correctly recognizes all prime numbers. Furthermore, the running time of every step is polynomial in $\log n$ (Exercise 4.5.8). It remains to clarify how the algorithm behaves when n is composite. After the encouraging examples from the exercises we might hope that composites will be correctly identified as such for any choice of base $a > 1$. Unfortunately this is false: for example, if the algorithm chooses $a = 2$ for the number $n = 2047 = 23 \cdot 89$, the output will incorrectly identify n as "probably prime". Similarly as for the Fermat test, we refer to such a number n as a **strong pseudo-prime** to base a. So we must deal with the following question: *if n is composite, how many numbers a are there with respect to which n is a strong pseudo-prime?*

For example, we could imagine that some Carmichael number n is also a strong pseudo-prime to every base a coprime to n. Thankfully this is not the case!

4.5.4. Theorem (No strong Carmichael numbers)**.**
Let $n = q \cdot r$, where $q, r > 2$ are odd and coprime. Then there is a number $a \in \mathbb{N}$ that is coprime to n and satisfies $a^2 \equiv 1$, but such that $a \not\equiv 1$ and $a \not\equiv -1 \pmod{n}$. In particular, n is not a strong pseudo-prime to base a.

Proof. By the Chinese Remainder Theorem (Theorem 3.1.7), we can find a number a that is coprime to n and satisfies $a \equiv -1 \pmod{q}$ and $a \equiv 1 \pmod{r}$. Then $a \not\equiv 1 \pmod{n}$, for otherwise we would also have $a \equiv 1 \pmod{q}$. For the same reason, $a \not\equiv -1 \pmod{n}$. On the other hand, we know that $a^2 \equiv 1 \pmod{q}$ and $a^2 \equiv 1 \pmod{r}$. By the Chinese Remainder Theorem, $a^2 \equiv 1 \pmod{n}$, as claimed.

Now write $n = d \cdot 2^\ell$ as in the Miller-Rabin primality test. Then $d = 2m + 1$ for suitable $m \geq 0$, and thus

$$a^d = \left(a^2\right)^m \cdot a \equiv a \not\equiv 1, -1 \quad \text{and} \quad a^{2d} = \left(a^2\right)^d \equiv 1 \pmod{n}.$$

By definition, n is not a strong pseudo-prime to base a. ■

Using this observation, we can conclude that, for any given composite number n, there must in fact be many suitable bases a.

4.5.5. Theorem (Error probability in MILLER-RABIN). *Let n be a composite number. Then there are at most $(n-1)/2$ bases $a \in \mathrm{cp}(n)$ to which n is a strong pseudo-prime.*

In other words, with probability at least 50%, the algorithm MILLER-RABIN correctly identifies n as composite.

Proof. Let W be the set of all numbers $a \in \mathrm{cp}(n)$ with respect to which n is a strong pseudo-prime. We would like to argue as in Lemma 3.3.3, using Lagrange's Theorem, that W contains no more than $\frac{\varphi(n)}{2} \leq \frac{n-1}{2}$ elements. But this is not quite possible, since the product of two numbers $a, b \in W$ need not again belong to W. Hence we need to be a little bit more careful.

First of all, the formulation of our algorithm implies that we may suppose that n is odd and not a perfect power. This means that n is of the form $n = q \cdot r$, where $q, r > 2$ are coprime. We again write $n - 1 = d \cdot 2^{\ell}$ and take a look at the largest index $j \leq l$ such that

$$a_0^{d \cdot 2^j} \not\equiv 1$$

for some $a_0 \in \mathrm{cp}(n)$. (Such a number $j \geq 0$ exists by the preceding theorem.) We define $k := d \cdot 2^j$ and study the set

$$G := \{a \in \mathrm{cp}(n) : a^k \equiv 1 \text{ or } a^k \equiv -1 \pmod{n}\}.$$

Every number $a \in W$ is also an element of G, but in contrast to W, the set G is closed under multiplication modulo n: if $a, b \in G$, then $a \cdot b \mod n$ is also an element of G. We now need to show that there is at least one number $a \in \mathrm{cp}(n)$ that is not an element of G. We construct such a number, quite similarly to Theorem 4.5.4, by using a_0 and the Chinese Remainder Theorem. First of all, if $a_0^k \not\equiv -1$, then $a_0 \notin G$ and we are done. Otherwise, the Chinese Remainder Theorem ensures that there is some number $a < n$ such that $a \equiv a_0 \pmod{q}$ and $a \equiv 1 \pmod{r}$. Then $a^k \equiv -1 \pmod{q}$ and $a^k \equiv 1 \pmod{r}$, and as in the proof of Theorem 4.5.4, we conclude that $a^k \not\equiv 1, -1 \pmod{n}$, and hence $a \notin G$.

Now we complete the proof just as in Lemma 3.3.3. We just showed that $\#G < \varphi(n)$, and Lagrange's Theorem (Theorem 3.2.10) tells us that $\#G$ divides $\varphi(n)$. So G contains at most $\varphi(n)/2$ elements, and thus

$$\#W \leq \#G \leq \frac{\varphi(n)}{2} \leq \frac{n-1}{2}. \qquad \blacksquare$$

We conclude that the algorithm MILLER-RABIN really is a Monte Carlo algorithm for compositeness. What is even better is that it essentially does not require any additional computations when compared to the Fermat test. After all, the computation of a^{n-1} using "divide and conquer" requires us to compute $a^d, a^{2d}, \dots$ anyway. Thus this algorithm is extremely well suited to practical applications – it is sufficiently fast to quickly reach a conclusion even for extremely large numbers. By running the algorithm several times, the error probability quickly becomes negligible (e.g. smaller than the probability of a hardware error).

We have reached the end of the first part of our book. In the second part, we will be able to contrast the Miller-Rabin algorithm with a *deterministic* efficient primality test. We would like to stress that – at least at this point in time – this new algorithm cannot hope to compete with the speed of the above method. The result is therefore mostly of *theoretical* interest.

Exercises.

4.5.6. Exercise. We recall from Section 3.3 that $n = 15$ is a pseudo-prime to base 11. This means that

$$a^{n-1} = 11^{14} \equiv 1 \mod n.$$

Compute the remainder of 11^7 after division by n and conclude that n is not a strong pseudo-prime to base 11. (So the Miller-Rabin test, with $a = 11$, recognizes $n = 15$ to be a composite number.)

4.5.7. Exercise (P). In this exercise, we show that $n = 561$, the first Carmichael number, will be recognized as composite by the Miller-Rabin algorithm, using the base $a = 2$. The calculations required can be carried out by hand with some dedication. However, it saves a lot of time to use a computer or calculator instead.

(a) Write $n - 1 = d \cdot 2^\ell$ as in Theorem 4.5.3.

(b) Compute the remainder b of 2^d after division by n.

(c) Compute b^2, b^4, and b^8 (modulo n).

(d) Conclude, using Theorem 4.5.3, that n is composite.

4.5.8. Exercise (!). For every step in the algorithm MILLER-RABIN, justify that the running time is at most polynomial in $\log n$.

Further Exercises and Comments.

4.5.9. By Exercises 3.3.8 and 3.3.9, even numbers and perfect powers are not Carmichael numbers. By Lemma 3.3.3, this implies that Theorem 4.5.5 remains correct if we remove Step **1** from the algorithm MILLER-RABIN.

4.5.10. Exercise. Use Lemma 4.5.1 to prove **Wilson's Theorem**: n is prime if and only if $(n-1)!+1$ is divisible by n. (Compare Exercise 3.1.9.)

4.5.11. A first version of the algorithm MILLER-RABIN was developed by Miller in 1976. This version was *deterministic*, not randomized. However, Miller was able to prove its correctness only by assuming the so-called **Generalized Riemann Hypothesis**, which remains unproven.

4.5.12. Solovay and Strassen designed the first efficient Monte-Carlo algorithm for compositeness in 1977. This algorithm is based on a different generalization of Fermat's Theorem, which uses the so-called **Legendre and Jacobi symbols** from number theory.

4.5.13. Rabin showed in 1980 how to modify Miller's method to obtain the Monte Carlo algorithm we described above. This test is usually more efficient than the algorithm of Solovay and Strassen.

In his paper, Rabin even showed that it is possible to replace the probability of 50% in Theorem 4.5.5 by the better bound of 75%. This requires some more careful analysis.

4.5.14. During the more than twenty years that passed between the development of the Miller-Rabin test and the breakthrough of Agrawal, Kayal, and Saxena, the field saw many other advances. Among the most important results of this period are:

(a) The development of Monte Carlo algorithms for *primality.* As described in Section 2.5, such methods can be combined with the Miller-Rabin algorithm to obtain *Las Vegas* algorithms for primality, which never return an incorrect result.

However, known methods that prove primality are not fast enough to keep up with the Miller-Rabin test and therefore are rarely used in practical applications.

(b) The development of *deterministic* algorithms for primality, whose running time, while not polynomial, is *subexponential*, in contrast e.g. to the Sieve of Eratosthenes. For example, since 1983 there have been methods known whose running time is at most

$$(\log n)^{O(\log\log\log n)}.$$

(This is not all that far off from a polynomial-time algorithm. For example, think about how large (or small) $\log\log\log n$ would be for prime numbers with several thousand digits, which is the order of magnitude used in cryptography.)

The mathematical methods and concepts that lead to these results are much deeper and more difficult than those we have encountered so far and also than those we will study in the second part of the book.

Further reading

For an introduction to cryptography, we recommend the book [**Be**] by Beutelspacher. Many aspects of number theory that are related to the topics of this chapter are beyond the scope of an introductory text. For the reader who would like to learn more about algorithmic and applied number theory, we recommend *Prime Numbers: A Computational Perspective* [**CP**] by Crandall and Pomerance; a lot of relevant interesting material can also be found in [**HW**].

Part 2

The AKS Algorithm

Chapter 5

The starting point: Fermat for polynomials

We are now ready to begin developing the primality test of Agrawal, Kayal, and Saxena, using the foundations laid in the first part of the book. As in the case of the Fermat test and the Miller-Rabin algorithm, the starting point will be a property of prime numbers, namely an extension of Fermat's Theorem for polynomials that we discuss in the first section of this chapter. In Section 5.2, we develop a strategy for turning this property into a deterministic and efficient primality test; we will use this strategy in the following chapters. The (optional) final section discusses a *randomized* primality test based on the same idea as the AKS algorithm.

5.1. A generalization of Fermat's Theorem

We begin with a generalization of Fermat's Theorem.

5.1.1. Theorem (Fermat for polynomials)**.**
Let p be a prime number. Then

$$(P(X))^p \equiv P(X^p) \pmod{p} \tag{5.1.2}$$

for all polynomials P with integer coefficients.

Examples. For $p = 3$ and $P = X + 1$,

$$(X+1)^3 = X^3 + 3X^2 + 3X + 1 \equiv X^3 + 1 \pmod{3}.$$

For $p = 5$ and $P = X - 1$, we similarly see that

$$(X-1)^5 = X^5 - 5X^4 + 10X^3 - 10X^2 + 5X - 1 \equiv X^5 - 1 \pmod{5}.$$

Before we prove Theorem 5.1.1, let us take a closer look at its statement. If $a \in \mathbb{Z}$ and $P = a$ is a constant polynomial, then the claim is that

$$a^p \equiv a \pmod{p}.$$

So Theorem 5.1.1 really does generalize Fermat's Theorem. At first glance, one might think that Fermat's Theorem also implies our new result. Indeed, by Fermat we have

$$(P(x))^p \equiv P(x) \equiv P(x^p) \pmod{p} \tag{5.1.3}$$

for all integers x. But we should remember that congruence of polynomials is defined using congruence of their coefficients, not of their values! So (5.1.3) is not equivalent to (5.1.2); recall Exercise 3.5.7. Hence, on the one hand, there certainly is something left to prove. On the other hand, for polynomials of degree 1, we could actually use Fermat's Theorem to prove our claim; see Exercise 5.1.14. (In fact, that is the only case that we will later need in the AKS algorithm.)

In the proof of Fermat's Theorem in Section 3.2, we used the fact that the binomial coefficient $\binom{p}{k}$ is divisible by p for all $k \in \{1, \dots, p\}$ (Lemma 3.2.4). The simple examples that we worked out above suggest that this will also be helpful for the proof of Theorem 5.1.1 because, after all, the same binomial coefficients appear when we multiply out $(P(X))^p$.

Proof of Theorem 5.1.1. In principle, we can multiply out the left-hand side of (5.1.2) and use Lemma 3.2.4 to prove that this agrees modulo p with the terms on the right-hand side. However, notation would quickly become confusing, since we have to deal with many different terms when multiplying out the p-th power of a polynomial of high degree. We avoid this problem by using induction on the degree d of the polynomial P.

If $d = 0$, then P is constant, and the claim follows from Fermat's Theorem. This establishes the basis of the induction.

For the inductive step, we suppose that the theorem holds for all integer polynomials of degree at most d. We need to prove the claim for a polynomial P of degree $d+1$. Let Q be the polynomial obtained by removing the highest term of P. That is, Q is a polynomial of degree at most d and there is some $a \in \mathbb{Z}$ such that

$$P = aX^{d+1} + Q.$$

Now we apply the binomial theorem:

$$\begin{aligned}(P(X))^p &= (aX^{d+1} + Q(X))^p \\ &= (aX^{d+1})^p + \left(\sum_{k=1}^{p-1} \binom{p}{k} (aX^{d+1})^k (Q(X))^{p-k}\right) + (Q(X))^p.\end{aligned}$$

We take a look at the terms in the last expression, one by one. First of all $\left(aX^{d+1}\right)^p = a^p\left(X^{(d+1)}\right)^p = a^p X^{p(d+1)} = a^p\left(X^p\right)^{d+1} \equiv a\left(X^p\right)^{d+1}$ modulo p, using Fermat's Theorem. Furthermore, by Lemma 3.2.4 the binomial coefficients $\binom{p}{k}$ in the sum are all divisible by p, so this sum is congruent to zero modulo p. Finally, $(Q(X))^p \equiv Q(X^p)$ modulo p by the induction hypothesis. So indeed

$$(P(X))^p \equiv a(X^p)^{d+1} + 0 + Q(X^p) = P(X^p) \pmod{p}.$$

This completes the induction and the proof of the theorem. ■

We can ask whether, in analogy to the *Carmichael numbers* from Section 3.3, there are also some composite numbers that satisfy (5.1.2) for *every* polynomial. It turns out that this is not the case. Indeed, if we try to apply the above proof to a composite number, we see that everything depends on the divisibility or non-divisibility of $\binom{n}{k}$ by n. A look at Pascal's triangle (Figure 3.3) suggests:

5.1.4. Lemma.
Let $n \geq 2$ be a natural number and let p be a prime divisor of n. Then $\binom{n}{p}$ is not divisible by n.

Proof. See Exercise 5.1.8(a). ■

We immediately deduce:

5.1.5. Theorem (Primality Criterion)**.**
Let $n \geq 2$ be a natural number, and let $a \in \mathbb{N}$ be coprime to n. Then n is prime if and only if

$$(X+a)^n \equiv X^n + a \pmod{n}. \tag{5.1.6}$$

Proof. If n is prime, then the claim follows from Theorem 5.1.1. For the converse, suppose that n is a composite number. Then n has some prime divisor $p < n$. By the binomial theorem, the p-th coefficient of $(X+a)^n$ is

$$a^{n-p}\binom{n}{p}.$$

As a is coprime to n by hypothesis and $\binom{n}{p}$ is not divisible by n because of Lemma 5.1.4, it follows that this coefficient is not divisible by n. The p-th coefficient of $X^n + a$ is zero, so we conclude that

$$(X+a)^n \not\equiv X^n + a \pmod{n}. \qquad \blacksquare$$

Remark. For simplicity's sake, we stated and proved the previous theorem only for polynomials of degree 1. However, it can be extended e.g. to polynomials of arbitrary degree whose coefficients are all coprime to n; see Exercise 5.1.16. So for many numbers n and polynomials P, the congruence

$$(P(X))^n \equiv P(X^n) \pmod{n} \tag{5.1.7}$$

holds if and only if n is prime.

At first, Theorem 5.1.5 seems like an incredibly strong result: we could test a number n for primality by choosing any number a coprime to n and checking the congruence (5.1.6)! Unfortunately this is, once again, impossible in practice. Indeed, we might have to compare up to n coefficients. So the effort required would be *exponential* in $\log n$, and thus of the same order of magnitude as searching directly for a factor of n or applying the Sieve of Eratosthenes.

Nonetheless it turns out that we can use Theorem 5.1.1 to formulate a deterministic and efficient primality test. The idea is to check the congruence (5.1.7) modulo a suitable polynomial Q of *small degree*. Then we can use the power algorithm to efficiently compute the power $(P(X))^n$, reducing the result modulo Q and n in every step. In this manner, we can efficiently test (5.1.7) modulo Q (for details, see Exercise 5.1.10). In Exercise 5.1.9, the reader has the opportunity to carry out this procedure by hand for a simple example.

So our strategy can be outlined as follows. We select, in some suitable manner, a polynomial P with integer coefficients as well as another polynomial Q, both of whose degrees are polynomially bounded in $\log n$. Then we test whether or not the congruence

$$(P(X))^n \equiv P(X^n) \pmod{n, Q}$$

holds. If not, we know because of Theorem 5.1.1 that n is not prime. Of course it might happen that some such congruences are also satisfied for composite numbers n; see Exercise 5.1.12. But it will turn out that, if we choose P and Q cleverly, we only need to check a fairly small number of congruences to detect a composite number. The next two chapters will show that this strategy really works.

Exercises.

5.1.8. Exercise (!). Let $n \geq 2$ be a natural number and let p be a prime divisor of n. Also let j be the largest exponent for which p^j divides n.

(a) Show that

$$\binom{n}{p} \not\equiv 0 \ (\mathrm{mod}\ p^j).$$

(*Hint:* As in Lemma 3.2.4, use the formula for binomial coefficients using factorials. Then consider how often the factor p occurs in each of the terms.)

(b) Also show that $\binom{n}{p^j} \not\equiv 0 \ (\mathrm{mod}\ p)$.

5.1.9. Exercise. Use the power algorithm to calculate

$$(X-1)^{24} \pmod{24, X^2-1}$$

by hand, and then check whether

$$(X-1)^{24} \equiv X^{24} - 1 \pmod{24, X^2-1}.$$

In contrast, try to calculate $(X-1)^{24}$ (mod 24) by hand.

5.1.10. Exercise (!). Let n be a natural number. Let Q and P be polynomials whose coefficients belong to the set $\{0, 1, \ldots, n-1\}$. Suppose that the degree r of Q is larger than the degree of P. Show that there is an algorithm that determines whether the congruence

$$(P(X))^n = P(X^n) \pmod{n, Q}$$

holds and whose running time is polynomial in r and $\log n$.

(If r is chosen in a way that grows at most polynomially with $\log n$, this means that the running time of this algorithm is also polynomial in $\log n$.)

5.1.11. Exercise (P). Implement the algorithm from Exercise 5.1.10.

5.1.12. Exercise. Show that

$$(X+1)^6 \equiv X^6 + 1 \pmod{6, X+3}.$$

Further Exercises and Comments.

5.1.13. In the article by Agrawal, Kayal, and Saxena, Theorem 5.1.1 is stated only for polynomials of degree one because this is sufficient for the algorithm. In order to appreciate the *idea* of the algorithm, however, it is useful to know the complete theorem, which dates back to the early nineteenth century. Similarly, Theorem 5.1.5 (and its generalization in Exercise 5.1.16) is helpful to motivate the development of the algorithm but will not be required in the proof later on.

5.1.14. Exercise. In this problem, we wish to prove Theorem 5.1.1, using only Fermat's Little Theorem, in the case where P has degree at most one. So let $P = aX + b$ with $a, b \in \mathbb{Z}$ and let p be a prime number.

(a) Use Theorem 3.2.2 to show that $(P(X))^p$ and $P(X^p)$ have the same leading coefficient. Conclude that the polynomial

$$R := (P(X))^p - P(X^p)$$

has degree at most $p-1$.

(b) Show, again using Theorem 3.2.2, that

$$R(x) \equiv 0 \pmod{p}$$

for all $x \in \mathbb{Z}$. Thus R has exactly p zeros modulo p.

(c) However, if R is not congruent to zero, then the number of its zeros modulo p is bounded by $\deg(R) \leq p-1$ (see Exercise 3.5.18). So we have shown that $R \equiv 0 \pmod{p}$, as claimed.

5.1.15. Exercise. Show that in Theorem 5.1.5 it is sufficient to assume that n does not divide a. (*Hint:* Use Exercise 5.1.8(b).)

5.1.16. Exercise. We study an extension of Theorem 5.1.5 to polynomials of arbitrary degree. Let n be a composite natural number, and let P be a polynomial for which each coefficient is either zero or coprime to n. Furthermore we require that P have at least two non-zero coefficients. Show that this implies that $(P(X))^n \not\equiv P(X^n) \pmod{n}$.

Idea for the proof: Similarly as in Theorem 5.1.1, choose $a \in \mathbb{Z}$ such that $P = aX^d + Q$. Let $d \geq 1$ denote the degree of P, let $m \geq 0$ be the degree of Q, and let k be the largest number between 1 and $n-1$ satisfying $\binom{n}{k} \not\equiv 0 \pmod{n}$. Show that the $(dK + m(n-k))$-th coefficient of $(P(X))^n$ is not divisible by n.

5.1.17. Exercise. Find a composite number n and natural numbers a, b between 1 and $n-1$ such that

$$(aX + b)^n \equiv aX^n + b \pmod{n}.$$

This shows that the condition of a and n being coprime in Exercise 5.1.16 cannot be completely removed.

5.2. The idea of the AKS algorithm

In order to develop an efficient primality test based on the vague idea we formulated in the previous section, we need to study congruences of the form

$$(P(X))^n \equiv P(X^n) \pmod{n, Q}, \tag{5.2.1}$$

where P and Q are polynomials and n is a composite number. In the following we abbreviate $R := (P(X))^n - P(X^n)$, so that (5.2.1) can be rewritten as $R \equiv 0 \pmod{n, Q}$.

We already saw in Chapter 3 that working modulo the composite number n is much more inconvenient than modulo a prime number, and it may seem that we cannot get around this if we are to study the above congruence. Fortunately, there is a trick to get us out of trouble: we instead study congruences modulo a prime factor p of n. Indeed, if we can (efficiently) find P and Q such that

$$R \not\equiv 0 \pmod{p, Q}, \tag{5.2.2}$$

then these also have the property that

$$R \not\equiv 0 \pmod{n, Q}.$$

This may seem somewhat strange. After all, we do not know the prime factor p when we wish to test n for primality. (Otherwise we already know that n is composite.) But that does not change the *mathematical* fact that such a prime factor exists, even if we do not explicitly know it; so we can use p in our analysis (but not in the formulation of the algorithm).

Nonetheless, our trick raises a question. We have seen before that, for any composite number n, there are always polynomials P such that $R \not\equiv 0 \pmod{n}$ (with our above notation). Is that still true if we replace n by a prime number p dividing n? Otherwise, we have a problem.

If we think back to the proof of Theorem 5.1.5, we quickly see that there are two different cases to consider. If n has two *different* prime divisors p and q, then indeed we have

$$(X + a)^n \not\equiv X^n + a \pmod{p}$$

for every number a coprime to n; see Exercise 5.2.4.

On the other hand, suppose that p is the *only* prime that divides n, so that $n = p^k$ for some $k > 1$. Then things look rather different. Indeed, we know from Theorem 5.1.1 that

$$(P(X))^{p^m} \equiv P(X^{p^m}) \pmod{p}$$

for all $m \geq 1$ (Exercise 5.2.3). In particular,

$$(P(X))^n \equiv P(X^n) \pmod{p},$$

and the congruence (5.2.2) cannot detect that n is composite.

Luckily, this does not pose a serious algorithmic obstacle. After all, we can test efficiently whether an integer is a perfect power (remember Exercise 2.3.6) and hence exclude such numbers right at the start of our algorithm. Then it only remains to consider composite numbers that have at least two different prime divisors, and for these our trick has the chance of being successful.

Next, we ask ourselves how we are going to proceed when choosing the polynomials P and Q. Here are two plausible alternatives:

(a) We *fix* some polynomial P as in Theorem 5.1.5 or Exercise 5.1.16, say $P = X - 1$. Then we test the congruence (5.2.1) for some *small* number of different polynomials Q.

(b) We first find some suitable polynomial Q and then test the congruence (5.2.1) for a *small* number of different polynomials P.

The procedure in (a) is motivated by the fact that, fixing P, we know that R is not congruent to zero modulo n (and also modulo p by Exercise 5.2.4). We would expect that we do not have to test too many different polynomials Q to find one that is not a divisor of R modulo p and hence does not satisfy (5.2.2).

In the next section, we indeed develop a simple *randomized* primality test on the basis of this idea, the *algorithm of Agrawal and Biswas*. Also, after our analysis of the AKS algorithm, we shall see in hindsight that approach (a) can indeed be used to formulate an efficient *deterministic* algorithm; see Exercise 7.2.4.

However, it turns out that approach (b) is easier to analyze mathematically. The reason is that studying varying congruences modulo a *fixed* polynomial is a lot easier than having to investigate a given congruence modulo many *different* polynomials. Therefore we shall use (b) to develop the AKS algorithm. We can now formulate the basic structure of the test as follows:

```
Basic structure of the AKS algorithm
Input:  A natural number $n > 1$.
  1. If $n$ is a perfect power, i.e. if $n = a^b$ with $a < n$
     and $b > 1$, then answer "$n$ is composite".
  2. Otherwise choose a "suitable" polynomial $Q$.
  3. Test the congruences
         $(P_i(X))^n \equiv P_i(X^n) \pmod{n, Q}$
     for "suitable" polynomials $P_1, \dots, P_\ell$.
  4. If one of these congruences is not satisfied, then
     answer "$n$ is composite".
  5. Otherwise answer "$n$ is prime".
     (Here $\ell$ and the degree of $Q$ are allowed to grow at
     most polynomially with $\log n$.)
```

Of course we still need to clarify a few things before we can turn this plan into a real algorithm.

(1) Let n be composite, but not a perfect power, and let p be a prime factor of n. If Q is a suitable polynomial, can we efficiently find a polynomial P such that

$$(P(X))^n \not\equiv P(X^n) \pmod{p, Q},$$

even without knowing p explicitly?

(2) For every $n \in \mathbb{N}$, is there such a "suitable" polynomial Q whose degree grows at least polynomially with $\log n$ and that can be found efficiently?

The first of these will be made precise and proved in the next chapter. Part (2) is treated in Chapter 7, completing our discussion of the AKS algorithm.

Exercises.

5.2.3. Exercise (!). Let p be a prime number and let $m \in \mathbb{N}$. Also let P be a polynomial with integer coefficients. Show, by using induction over m and applying Theorem 5.1.1, that

$$(P(X))^{p^m} \equiv P(X^{p^m}) \pmod{p}.$$

5.2.4. Exercise (!). Let n be a composite number that has at least two different prime divisors p and q. Show that

$$(X+a)^n \not\equiv X^n + a \pmod{p}$$

for all integers a that are coprime to n. (*Hint*: Exercise 5.1.8(b).)

Further Exercises and Comments.

5.2.5. Knowing the work of Agrawal, Kayal, and Saxena, it is not hard to find good arguments for the approach explained in (b). However, when pursuing mathematical research, and without the benefit of hindsight, it is much more difficult to decide which approaches are promising. Hence it is crucial to keep an open mind about trying alternative methods.

Indeed, Agrawal and his students did *not* start with approach (b) but rather attempted to make (a) work. Over time, they decided to modify their strategy, leading to their eventual success.

5.3. The Agrawal-Biswas test

We now use Theorem 5.1.5 to develop a randomized Monte Carlo algorithm for the problem COMPOSITES. (We already know one such algorithm from Section 4.5, namely the Miller-Rabin test.) This algorithm was presented in 1999 by Manindra Agrawal and his former PhD advisor, Somenath Biswas (the article was published in 2003 [**AB**]). Even though this primality test cannot compete with the Miller-Rabin algorithm in practical terms, it indicates that Theorem 5.1.5 can be used to efficiently test for primality. Historically, it was the first step that lead to the development of the AKS algorithm. We emphasize that the results presented here will not be used in the remaining chapters.

The idea of the algorithm is very simple: we choose an arbitrary polynomial P as in Theorem 5.1.5 or Exercise 5.1.16; for simplicity, we will use $P = X - 1$. Then we (randomly) pick a polynomial Q of degree $q := \lceil \log n \rceil$ and check the congruence

$$(P(X))^n - P(X^n) \equiv 0 \pmod{n, Q}.$$

To show that this gives a Monte Carlo algorithm, we need to give a lower bound for the probability that, for a composite number $n \in \mathbb{N}$, this congruence is satisfied for a randomly chosen polynomial Q. As explained in the last section, it is reasonable to study these congruences modulo a prime divisor p of n rather than modulo n itself.

So let $n \in \mathbb{N}$ again be a composite number that is not a prime power, and let p be a prime divisor of n. Let $\mathcal{Q}$ denote the set of all monic polynomials of degree $q := \lceil \log n \rceil$ with integer coefficients between 0 and $p - 1$. (We fix this notation for the remainder of this section.) Then we ask how many different polynomials in $\mathcal{Q}$ could divide the polynomial

$$R := (P(X))^n - P(X^n)$$

modulo p. (We know from Exercise 5.2.4 that $R \not\equiv 0 \pmod{p}$.) We shall use, without proof, the following theorem [**LiN**, Corollary 3.12] regarding the number of irreducible polynomials.

5.3.1. Theorem (Number of irreducible polynomials modulo p)**.** *There are at least $\frac{p^q}{q} - p^{q/2}$ polynomials in $\mathcal{Q}$ that are irreducible modulo p.*

Remark. This theorem should be viewed as a generalization of the prime number theorem. Indeed, the number of polynomials in $\mathcal{Q}$ is exactly $N := p^q$. Thus the theorem states that the number of irreducible elements in $\mathcal{Q}$ grows asymptotically at least as fast as $N/\log N$.

5.3.2. Corollary (Number of irreducible polynomials in $\mathcal{Q}$)**.** *If $p \geq 5$, then there are at least $\frac{p^q}{2q}$ polynomials in $\mathcal{Q}$ that are irreducible modulo p.*

Proof. By Theorem 5.3.1, the number of polynomials in $\mathcal{Q}$ that are irreducible modulo p is at least $\frac{p^q}{q} - p^{q/2}$. We can rewrite this number as

$$\frac{p^q}{q} \cdot \left(1 - \frac{q}{p^{q/2}}\right).$$

The fraction $q/p^{q/2}$ becomes very small if p is sufficiently large. More precisely, we must show that

$$\frac{q}{p^{q/2}} < \frac{1}{2}$$

when $p \geq 5$. But this follows immediately from the fact that

$$p^{q/2} > 4^{q/2} = 2^q \geq 2q.$$

Here the final inequality is taken from Exercise 1.1.12(a). ■

We now know that $\mathcal{Q}$ contains quite a lot of irreducible polynomials. However, at most n/q of these could be divisors of R modulo p. Indeed, since p is prime, R can be decomposed uniquely into factors that are irreducible modulo p (Exercise 3.5.16). The sum of the degrees of these factors is exactly the degree of R, and hence at most n. So the number of irreducible factors of R of degree q is at most n/q, as claimed. We use this to deduce:

5.3.3. Lemma (Many polynomials do not divide R)**.**
Let $p \geq 5$. Then there are at least $p^q/4q$ polynomials in $\mathcal{Q}$ that do not divide R modulo p.

Proof. By Corollary 5.3.2, there are at least $p^q/2q$ polynomials in $\mathcal{Q}$ that are irreducible modulo p. At most n/q of these divide R. So the number of polynomials that we are looking for is at least

$$\frac{p^q}{2q} - \frac{n}{q} = \frac{p^q - 2n}{2q} = \frac{2p^q - 4n}{4q}.$$

Hence it remains to show that $p^q \geq 4n$. To see this, we remember that $q = \lceil \log n \rceil$ and $n \geq p > 4$; hence

$$p^q \geq p^{\log n} = n^{\log p} > n^2 > 4n,$$

and the proof is complete. ■

5.3.4. Corollary (Probability of finding a non-divisor)**.**
Let $p \geq 5$ be a prime and let Q be a randomly chosen monic polynomial of degree q with integer coefficients between 0 and $n-1$. Then the probability that Q does not divide R modulo p is at least

$$\frac{1}{4q}.$$

Proof. Let $\mathcal{Q}'$ denote the set of monic polynomials of degree q with integer coefficients between 0 and $n-1$. Then $\mathcal{Q}'$ contains exactly n^q different polynomials. Each of these is picked with the same probability, so the probability of randomly choosing any given polynomial is $1/n^q$.

By Lemma 5.3.3 there are at least $p^q/4q$ polynomials in $\mathcal{Q}$ that do not divide R modulo p. Every polynomial in $\mathcal{Q}$ is congruent modulo p to exactly $(n/p)^q$ different polynomials from $\mathcal{Q}'$. (We leave it to the reader to work out why!)

Hence $\mathcal{Q}'$ contains at least $(p^q/4q) \cdot (n/p)^q = n^q/4q$ polynomials that do not divide R modulo p. So the probability of picking one of these is at least

$$\frac{n^q}{4q} \cdot \frac{1}{n^q} = \frac{1}{4q},$$

as claimed. ■

Now we are ready to formulate the Agrawal-Biswas test.

```
ALGORITHM OF AGRAWAL AND BISWAS
Input:  A natural number $n \geq 2$.
  1. If $n = 2$ or $n = 3$, answer "$n$ is prime".
  2. If $n$ is divisible by 2 or 3, answer "$n$ is composite".
  3. If $n$ is a perfect power, answer "$n$ is composite".
  4. Otherwise choose, randomly, a monic polynomial
     $Q$ of degree $q = \lceil \log n \rceil$ and with integer coefficients
     between 0 and $n - 1$.
  5. Check whether
         $(X-1)^n \equiv X^n - 1 \pmod{n, Q}$.
  6. If not, then answer "$n$ is composite". Otherwise,
     answer "$n$ is probably prime".
```

The first two steps can clearly be carried out efficiently. If n passes both, then the smallest prime divisor of n is at least 5 and Corollary 5.3.4 is applicable to n. The third step – testing whether n is a perfect power – can also be performed efficiently (Exercise 2.3.6).

For the fourth step, we randomly choose q integers between 0 and $n - 1$ to be the coefficients of the polynomial Q, with leading coefficient 1 because Q is monic. As $q = \lceil \log n \rceil$, this can be done efficiently. We already saw in Exercise 5.1.10 that Step **5** can also be done efficiently. So the algorithm of Agrawal and Biswas is efficient.

If n is composite, then, by Corollary 5.3.4, the probability that the algorithm answers "n is composite" is at least $1/(4\lceil \log n \rceil)$. As with the random generation of prime numbers, it is important that this probability is at worst polynomial in $1/\log n$. Indeed, if n is composite, we only have to run the algorithm on average $4\lceil \log n \rceil$ times to detect this. (See Exercise 2.5.6.) So if we carry out the algorithm of Agrawal and Biswas $8\lceil \log n \rceil$ times, then the probability that n is detected as a composite number is at least 50% (Exercise 2.5.10). So we have indeed found a Monte Carlo algorithm for compositeness.

Remark. With a little more effort it is possible to show that the probability of error in the algorithm of Agrawal and Biswas is at most 1/3 (without having to run the algorithm several times), which of course is a lot better than our bound of $1/4\lceil \log n \rceil$. The details of this argument can be found in the article [**AB**].

Chapter 6

The theorem of Agrawal, Kayal, and Saxena

We now turn our attention to item (1) from Section 5.2. So let n be a natural number that is not a power of a prime and let p be a prime factor of n. We must show that, for a "suitable" polynomial Q, we can efficiently find a polynomial P such that

$$(P(X))^n \not\equiv P(X^n) \pmod{p, Q}.$$

The idea is to show that the set $\mathcal{P}$ of polynomials P for which $(P(X))^n$ and $P(X^n)$ *are* congruent modulo p and Q is in some sense "not too large" and that, in particular, we can efficiently find a polynomial that violates this congruence. We formulate a precise version of this statement, the **theorem of Agrawal, Kayal, and Saxena**. In particular, we shall fix our choice of the polynomial Q. Section 6.2 discusses the idea underlying the proof, while the main work is done in Section 6.3. This will give us an estimate on the number of polynomials of degree 1 that belong to $\mathcal{P}$.

To conclude, Section 6.4 investigates the irreducible factors of **cyclotomic polynomials** modulo a prime number p. This is necessary in order to deduce the theorem of Agrawal, Kayal, and Saxena, as formulated in Section 6.1, from the results of Section 6.3.

6.1. Statement of the theorem

The goal of this chapter is to prove the following theorem, which tells us how to find a polynomial that identifies n as composite, provided that n is not a prime power.

6.1.1. Theorem (Theorem of Agrawal, Kayal, and Saxena)**.**
Let $r \in \mathrm{cp}(n)$ be a prime number with $\mathrm{ord}_r(n) > 4(\log n)^2$. Also set $Q := X^r - 1$. If n is not a power of p, then there are at most r polynomials of the form $P = X + a$, with $a \in \{0, \dots, p-1\}$, that satisfy

$$(P(X))^n \equiv P(X^n) \pmod{p, Q}. \tag{6.1.2}$$

Let us first establish that this theorem, once proven, really gives us what we are looking for. Indeed, suppose that we can find a number r satisfying the hypothesis of the theorem such that r grows at most polynomially in $\log n$. Then we need to check only for at most r different integers a whether the congruence (6.1.2) is satisfied. If this congruence always holds, then n is a prime or a prime power; otherwise n is composite. (We will see in the next chapter that it is indeed possible to find such a number r.)

Why are we using $Q = X^r - 1$? One of the reasons is that a congruence modulo Q and n implies that many other congruences also hold, as shown in the following lemma. This turns out to be very useful in the proof of Theorem 6.1.1.

6.1.3. Lemma.
Let $r \in \mathbb{N}$ and let $Q := X^r - 1$, $m \in \mathbb{N}$, and $n \geq 2$. Then

(a) *$Q(X^m) = X^{rm} - 1 \equiv 0 \pmod{Q}$; and*

(b) *if P is a polynomial such that $P \equiv 0 \pmod{n, Q}$, then also*

$$P(X^m) \equiv 0 \pmod{n, Q}.$$

Proof. The first statement simply claims that $X^r - 1$ is a divisor of $X^{rm} - 1$, which follows from the following calculation:

$$X^{rm} - 1 = (X^r - 1)(X^{(m-1)r} + X^{(m-2)r} + \cdots + X^r + 1).$$

(This expression can be found e.g. by using polynomial long division.)

Now let n and P be given as in (b). The assumption means that Q is a divisor of P modulo n. Hence there is a polynomial R such that $P \equiv R \cdot Q \pmod{n}$. If we substitute X^m for X in this congruence, then we see that, in particular,

$$P(X^m) \equiv R(X^m)Q(X^m) \pmod{n}.$$

So the polynomial $Q(X^m)$ divides $P(X^m)$ modulo n. The first part of the lemma implies that Q is a divisor of $Q(X^m)$, and hence Q divides $P(X^m)$ modulo n, as claimed. ■

Further Exercises and Comments.

6.1.4. The original article by Agrawal, Kayal, and Saxena proves a slightly stronger version of Theorem 6.1.1 than the one we stated. In particular, "at most r" in the statement can be replaced by "at most $2\sqrt{r}\log n$"; see also Exercise 6.3.9. Furthermore it is not necessary to require that r be prime; see Comment 6.4.13. We also refer the reader to Section 7.3.

6.2. The idea of the proof

We begin by fixing some notation for the upcoming discussion.

6.2.1. Notation.
For the remainder of this chapter, let $n \geq 2$, let p a prime divisor of n, and fix some number $r \in \mathbb{N}$. Define $Q := X^r - 1$ and let $\mathcal{P}$ denote the set of all integer polynomials P (of any degree) satisfying (6.1.2).

By Theorem 5.1.1, we know that every polynomial $P \in \mathcal{P}$ satisfies not only (6.1.2) but also

$$P(X^p) \equiv (P(X))^p \pmod{p, Q}.$$

Our choice of Q allows us to say much more:

6.2.2. Lemma.
Let P be a polynomial, and let m_1 and m_2 be natural numbers such that

$$(P(X))^{m_1} \equiv P(X^{m_1}) \quad \textit{and} \quad (P(X))^{m_2} \equiv P(X^{m_2}) \pmod{p, Q}.$$

If we set $m := m_1 \cdot m_2$, then also

$$(P(X))^m \equiv P(X^m) \pmod{p, Q}. \tag{6.2.3}$$

Proof. We have $(P(X))^{m_1} \equiv P(X^{m_1})$ modulo p and Q. Using Lemma 6.1.3, we see that

$$(P(X^{m_2}))^{m_1} \equiv P(X^{m_1 \cdot m_2}) \pmod{p, Q}.$$

Since furthermore $(P(X))^{m_2} \equiv P(X^{m_2})$, we conclude that

$$(P(X))^{m_1 \cdot m_2} = ((P(X))^{m_2})^{m_1} \equiv (P(X^{m_2}))^{m_1} \equiv P(X^{m_1 \cdot m_2})$$

modulo p and Q, as claimed. ■

6.2.4. Corollary.
Let $P \in \mathcal{P}$. Then (6.2.3) *holds for every number m of the form $m = n^i \cdot p^j$, where $i, j \geq 0$.*

Proof. As remarked above, both p and n satisfy the condition (6.1.2). The claim follows by repeated application of the preceding lemma. ■

Thus, if n is not a power of p, then there are many numbers m such that (6.2.3) holds for all polynomials in $\mathcal{P}$. This is a strong condition, and Theorem 6.3.6 below will show that the set $\mathcal{P}$ cannot be too large. Moreover $\mathcal{P}$ has the rather nice property that we can combine elements from $\mathcal{P}$ to build new ones.

6.2.5. Lemma.
Let $P_1, P_2 \in \mathcal{P}$. Then also $P_1 \cdot P_2 \in \mathcal{P}$.

Proof. From the definition of $\mathcal{P}$ it follows immediately that

$$\begin{aligned}(P_1 \cdot P_2)(X^n) &= P_1(X^n) \cdot P_2(X^n) \equiv (P_1(X))^n \cdot (P_2(X))^n \\ &= ((P_1 \cdot P_2)(X))^n \pmod{p, Q}.\end{aligned}$$ ■

Therefore, if we check (6.1.2) for a certain number of polynomials, we have actually proved this congruence for a much larger set. If there are many polynomials of degree 1 in $\mathcal{P}$, then Lemma 6.2.5 provides us with the existence of many more elements of $\mathcal{P}$ of small degree. (We make this precise in Theorem 6.3.5.)

As mentioned above, we will also establish an upper bound on the size of $\mathcal{P}$, provided that n is not a prime power. This upper bound, together with the observation we have just made, yields a restriction on the number of degree 1 polynomials that can belong to $\mathcal{P}$, proving Theorem 6.1.1.

Further Exercises and Comments.

6.2.6. In their article, Agrawal, Kayal, and Saxena use the term "introspective numbers" for those numbers m that satisfy (6.2.3).

6.3. The number of polynomials in $\mathcal{P}$

After all our preliminary deliberations, we are now ready to do the main work for the proof of Theorem 6.1.1. We begin by adjusting our point of view a little. In the previous section, we always calculated modulo Q. But the polynomial Q is not irreducible modulo p (for example, $X - 1$ is one of its divisors), and modular arithmetic with respect to such a polynomial, similarly as with respect to composite numbers, quickly becomes rather complicated.

But we already know how to get around this: in Section 5.2, we decided to carry out our analysis modulo p instead of working modulo n, and the same idea applies here. By Exercise 3.5.15, there is some irreducible factor H of Q modulo p. (We study the properties of such H in the following section.) Then

$$(P(X))^n \equiv P(X^n) \pmod{p, H} \tag{6.3.1}$$

for every $P \in \mathcal{P}$. So we will fix the irreducible factor H from now on and perform our considerations modulo H.

How many polynomials are there in $\mathcal{P}$ that are pairwise different modulo p and H? Let us denote this number by A. The number is certainly finite: after all, every integer polynomial is congruent (modulo p and H) to one whose coefficients range between 0 and $p-1$ and whose degree is smaller than that of H. So $A \leq p^{\deg(H)}$.

As announced in the previous section, we will now deduce both upper and lower bounds for A. To do so, we need to decide when two elements of $\mathcal{P}$ are congruent to each other modulo p and H. Thankfully this is not too difficult. The following lemma shows that two polynomials of sufficiently small degree that are different modulo p will also not be congruent modulo p and H.

In the lemma and in the remainder of this chapter, we let t denote the number of polynomials of the form $X^{n^i \cdot p^j}$ with $i, j \geq 0$ that are different modulo p and H.

6.3.2. Lemma.
Let P_1 and P_2 be polynomials in $\mathcal{P}$ whose degree is smaller than t. If $P_1 \equiv P_2 \pmod{p, H}$, then also $P_1 \equiv P_2 \pmod{p}$.

Example. We consider the case $n = 38$, $p = 19$, $r = 5$. The polynomial $Q = X^5 - 1$ has the following irreducible factors modulo p:

$$(6.3.3)\quad X^5 - 1 \equiv (X-1)\cdot(X^2+5X+1)\cdot(X^2-4X+1) \pmod{19}.$$

Indeed, the congruence is easily checked by a simple calculation. The two factors of degree 2 have no zeros modulo p and therefore have no factors of degree 1. Thus they are really irreducible. (How to go about finding the factors in (6.3.3) is another question!)

We set $H := X^2 + 5X + 1$ and determine the number t. First of all we see that $X^5 \equiv 1 \pmod{p, H}$ because H divides the polynomial $X^5 - 1$. This implies that X^{m_1} and X^{m_2} are congruent modulo p and H whenever m_1 and m_2 are congruent modulo 5, and thus $t \leq 5$. Furthermore, modulo p and H,

$$X^0 = 1; \qquad X^p = X^{19} = (X^5)^3 \cdot X^4 \equiv X^4 \equiv 18X + 14;$$

$$X^n = X^{38} = (X^5)^7 \cdot X^3 \equiv X^3 \equiv 5X + 5;$$

$$X^{p\cdot n} = X^p \cdot X^n \equiv X^3 \cdot X^4 \equiv X^2 \equiv 14X + 18; \qquad X^{p^2} \equiv X^{16} \equiv X.$$

(In each case we performed a polynomial long division at the end, if necessary, to obtain a polynomial that has smaller degree than H.)

We see that these five polynomials are pairwise not congruent, so it follows that $t = 5$. Now if two polynomials from $\mathcal{P}$ are not congruent modulo p and have degree at most 4, then we know by the lemma that they are not congruent modulo p and H. This is very useful – the point is that the number t will, in general, be larger than the degree of H.

Proof of Lemma 6.3.2. By Corollary 6.2.4 and by hypothesis,

$$P_1(X^m) \equiv (P_1(X))^m \equiv (P_2(X))^m \equiv P_2(X^m) \pmod{p, H}$$

for every number m of the form $m = p^j \cdot n^i$. In other words, for every such m the polynomial X^m is a *polynomial zero* of

$$T := P_1(Y) - P_2(Y)$$

modulo p and H (in the sense discussed in Section 3.5). By definition of t, the polynomial T has at least t different polynomial zeros modulo p and H. On the other hand, the degree of T is less than t (by our hypothesis on P_1 and P_2). Hence Theorem 3.5.6 forces $T \equiv 0 \pmod{p}$ as claimed. ■

If $\mathcal{P}$ contains many polynomials of degree 1, then we can use the preceding lemma to obtain a good lower bound for the number A. To do so, we define:

6.3.4. Definition.
In the following, let ℓ denote the number of elements $a \in \mathbb{N}_0$ with $a \leq p-1$ for which the polynomial $X + a$ is an element of $\mathcal{P}$.

We shall soon see that ℓ cannot be too large unless n is a prime power. As preparation for our next results, we remind ourselves:

- H is an irreducible factor of Q modulo p;
- A is the number of elements of $\mathcal{P}$ that are pairwise different modulo p and H;
- t is the number of polynomials of the form $X^{n^i \cdot p^j}$, where $i, j \geq 0$, that are pairwise different modulo p and H.

6.3.5. Theorem (Lower bound on A in terms of t and ℓ)**.**
The number A satisfies $A \geq \binom{t+\ell-1}{t-1}$.

Proof. We already mentioned the idea in the previous section. Let $k < t$ and let $a_1, \ldots, a_k \in \{0, \ldots, p-1\}$ be such that $X+a_1, \ldots, X+a_k$ are elements of $\mathcal{P}$. The product of these is a polynomial of degree less than t that also belongs to $\mathcal{P}$, by Lemma 6.2.5. Using Exercise 3.5.11, we know that the numbers $a_1, \ldots, a_k$, up to reordering, are uniquely determined by the polynomial T.

If $k' < t$ and $b_1, \ldots, b_{k'} \in \{0, \ldots, p-1\}$ are chosen such that $X + b_1, \ldots, X + b_{k'} \in \mathcal{P}$, then it follows that either the coefficients a_i and b_j agree up to reordering or the product $T' := \prod\limits_{1 \leq j \leq k'} (X + b_j)$ is not congruent to T modulo p. In the latter case, we see from Lemma 6.3.2 that T and T' are also not congruent modulo p and H.

Thus the only remaining question is: starting from a set of ℓ different polynomials of degree 1 and taking products, how many polynomials of degree at most t can we obtain? In other words, given ℓ balls, how many possibilities are there to pick up to t, not necessarily different, balls if the order is irrelevant? By Exercise 1.1.18 the answer is exactly $\binom{t+\ell-1}{t-1}$ and thus the theorem is proved! ■

As promised, we can complement this *lower* bound by an *upper* bound that holds if n is not a power of p.

6.3.6. Theorem.
If n is not a power of p, then $A \leq n^{2\sqrt{t}}/2$.

Proof. We take a look at numbers of the form $m = n^i \cdot p^j$. If n is not a power of p, then different choices of i and j will result in a different number m. If we require that also $0 \leq i, j \leq \lfloor\sqrt{t}\rfloor$, then there are exactly $(\lfloor\sqrt{t}\rfloor + 1)^2 > t$ such choices.

By definition of t, we can thus find two numbers of that form, let us call them m_1 and m_2, for which

$$X^{m_1} \equiv X^{m_2} \pmod{p, H}. \tag{6.3.7}$$

We choose notation such that $m_1 > m_2$. Then

$$m_2 < m_1 \leq (np)^{\lfloor\sqrt{t}\rfloor} \leq \frac{n^{2\sqrt{t}}}{2}.$$

Here we used that p is a non-trivial divisor of n and hence $p \leq n/2$. For all polynomials $P \in \mathcal{P}$, it follows from congruence (6.3.7) and Corollary 6.2.4 that

$$(P(X))^{m_1} \equiv P(X^{m_1}) \equiv P(X^{m_2}) \equiv (P(X))^{m_2} \pmod{p, H}.$$

Therefore every element $P \in \mathcal{P}$ is a polynomial zero of

$$R := Y^{m_1} - Y^{m_2}$$

modulo p and H. The polynomial R has degree m_1 modulo p; in particular, $R \not\equiv 0 \pmod{p}$. By Theorem 3.5.6, the number of polynomial zeros of R modulo p and H is bounded from above by m_1. This means that

$$A \leq m_1 \leq \frac{n^{2\sqrt{t}}}{2}. \qquad ■$$

If n is not a power of p, then we can now bound A both from above and from below. We notice that the lower bound, namely the binomial coefficient

$$\binom{t+\ell-1}{t-1},$$

will grow exponentially in t if ℓ is sufficiently large, whereas the upper bound is exponential in $\sqrt{t} \cdot \log n$. If we can make sure that t is significantly larger than $(\log n)^2$, then the upper bound is smaller than the lower bound, and we have a contradiction! We work out the details in the next corollary.

6.3.8. Corollary.
If $t > 4(\log n)^2$ and $\ell \geq t-1$, then n is a power of p.

Proof. By hypothesis and Theorem 6.3.5, we see that

$$A \geq \binom{t+\ell-1}{t-1} \geq \binom{2(t-1)}{t-1} \geq 2^{t-1} = \frac{2^t}{2},$$

using the inequalities for binomial coefficients from Exercise 1.1.17. So A grows exponentially in t, as mentioned above.

On the other hand, the first hypothesis says that $\sqrt{t} > 2\log n$. In particular

$$t = \sqrt{t} \cdot \sqrt{t} > 2\sqrt{t}\log n.$$

Then

$$\frac{2^t}{2} > \frac{2^{2\sqrt{t}\log n}}{2} = \frac{n^{2\sqrt{t}}}{2}$$

and thus overall $A > n^{2\sqrt{t}}/2$. By Theorem 6.3.6 this is possible only when n is a power of p. ■

To make use of Corollary 6.3.8, we still need to get a handle on the mysterious number t, which depends on the irreducible factor H of $X^r - 1$. We deal with this in the next section.

Exercises.

6.3.9. Exercise. Show that the hypothesis $\ell \geq t - 1$ in Corollary 6.3.8 can be replaced by the weaker condition $\ell > 2\sqrt{t}\log n$.

Further Exercises and Comments.

6.3.10. Exercise. Can the idea of the proof of Theorem 6.3.6 also give an upper bound for the number A when n *is* a prime power? If so, what is the order of magnitude of this bound, and why does this not lead to a contradiction when combined with Theorem 6.3.5?

6.3.11. We remark that, throughout this and the previous section, we did *not* assume that r is prime. See also Comment 6.4.13.

6.4. Cyclotomic polynomials

Finally, we need to take a closer look at the irreducible factors of the polynomial $X^r - 1$ modulo p. This is particularly easy when r is a prime number, so from now on let us suppose that this is the case. We write $X^r - 1$ as

$$X^r - 1 = (X - 1) \cdot (X^{r-1} + X^{r-2} + \cdots + X + 1)$$

(which can be seen directly by multiplying out). The polynomial

$$K_r := X^{r-1} + X^{r-2} + \cdots + X + 1 \tag{6.4.1}$$

is called the r**-th cyclotomic polynomial**.

K_r is an irreducible factor of $X^r - 1$ over $\mathbb{Z}$ (see Exercise 6.4.5). But we are interested in irreducible factors *modulo a prime number* p, and in general K_r is not irreducible modulo p. We thus have to study the factors of K_r modulo p.

6.4.2. Lemma.
Let p and r be prime numbers with $p \neq r$, and let H be an irreducible factor (modulo p) of the r-th cyclotomic polynomial K_r. Then

$$X^r \equiv 1 \pmod{p, H} \quad \textit{and}$$
$$X^k \not\equiv 1 \pmod{p, H}$$

for all $k \in \{1, \ldots, r-1\}$.

Proof. We have $X^r \equiv 1 \ (\mathrm{mod}\ X^r - 1)$ and hence also

$$X^r \equiv 1 \pmod{p, H}$$

because H is a divisor of $X^r - 1$ modulo p. Now let $k \geq 1$ be the smallest number with

$$X^k \equiv 1 \pmod{p, H}.$$

Then k is a divisor of r (Exercise 6.4.4). Since r is prime, we must have $k = r$ or $k = 1$.

Assume, by way of contradiction, that $k = 1$. This means that $X - 1 \equiv 0 \ (\mathrm{mod}\ p, H)$, and since H is irreducible modulo p, it follows that $H \equiv X-1 \ (\mathrm{mod}\ p)$. So $X-1$ is a divisor of K_r modulo p; in other words, $X - 1 \equiv 0 \ (\mathrm{mod}\ p, K_r)$. In particular, $K_r(1) \equiv 0 \ (\mathrm{mod}\ p)$. But we also know that

$$K_r(1) = 1 + 1 + \cdots + 1 = r \not\equiv 0 \pmod{p},$$

which is a contradiction. So $k = r$ as desired. ■

6.4.3. Corollary.
Let r and p be prime numbers with $r \neq p$ and let H be an irreducible factor (modulo p) of the r-th cyclotomic polynomial K_r. Also let $n \in \mathbb{N}$ be any multiple of p such that $\gcd(n, r) = 1$, and let t be as defined in Section 6.3. Then $\mathrm{ord}_r(n) \leq t \leq r$.

Proof. We remind ourselves that t denotes the number of polynomials of the form X^m that are different modulo p and H, where m ranges over all products of powers of n and p.

The order $\operatorname{ord}_r(n)$ is, by definition, exactly the number of pairwise different elements of the form $m = n^j$ modulo r. If m_1 and m_2 are different powers of n modulo r, then also $X^{m_1} \not\equiv X^{m_2} \pmod{p, H}$ by Lemma 6.4.2. This proves the first inequality.

By Lemma 6.4.2 there are overall at most r polynomials of the form X^m that are different modulo p and H, no matter how m is chosen. This proves the second inequality. ■

Proof of Theorem 6.1.1. Let n and p be as before, i.e. p is prime and n is a multiple of p. Moreover let r be a prime number coprime to n and satisfying $\operatorname{ord}_r(n) > 4(\log n)^2$. Since r is coprime to n, we know that $r \neq p$, and hence

$$4(\log n)^2 < t \leq r$$

by Corollary 6.4.3. As in the previous section, let ℓ denote the number of integers $a \in \{0, \ldots, p-1\}$ for which $X+a$ satisfies (6.1.2). Suppose that $\ell \geq r$. Then $\ell \geq t \geq t-1$, and Corollary 6.3.8 implies that n is a power of p. This proves the theorem. ■

Exercises.

6.4.4. Exercise (!). Let $n \geq 2$ and let H be a monic polynomial. (More generally, we could let H be a polynomial whose leading coefficient is coprime to n, so that we can divide by H with remainder.) Suppose that we are given $r \geq 1$ with $X^r \equiv 1 \pmod{n, H}$. Show that if k is the smallest natural number satisfying $X^k \equiv 1 \pmod{n, H}$, then k divides r.

6.4.5. Exercise. Let r be prime and let K_r be as in (6.4.1).

(a) We define a polynomial K' by substituting $X+1$ for X in K_r; i.e. $K' := K_r(X+1)$. Prove that

$$K' \equiv X^{r-1} \pmod{r}.$$

Hint: By definition of K_r, we see that $X \cdot K' = (X+1)^r - 1$. Use Theorem 5.1.1 to compute this polynomial modulo r.

(b) Use *Eisenstein's irreducibility criterion* (Exercise 3.5.20) to show that K' is irreducible over $\mathbb{Q}$ and $\mathbb{Z}$.

(c) Conclude that K_r is also irreducible.

Further Exercises and Comments.

6.4.6. The r-th cyclotomic polynomial can be defined for all natural numbers r. However, for composite numbers the definition is not given by (6.4.1). Instead, K_r is defined to be the polynomial that remains when dividing $X^r - 1$ by the product of all cyclotomic polynomials K_m for which $m < r$ is a divisor of r. In other words, K_m is defined inductively by the equation

$$X^r - 1 = \prod_{m \in \mathbb{N}, m | r} K_m. \tag{6.4.7}$$

For example,

$$\begin{aligned} X^6 - 1 &= (X-1)(X+1)(X^2+X+1)(X^2-X+1) \\ &= K_1 \cdot K_2 \cdot K_3 \cdot (X^2 - X + 1), \end{aligned}$$

so $X^2 - X + 1$ is the sixth cyclotomic polynomial.

The following exercises and comments are concerned with properties of cyclotomic polynomials.

6.4.8. Exercise. Compute the cyclotomic polynomial K_r for all $r \leq 10$.

6.4.9. Exercise.

(a) Show that K_r has degree $\varphi(r)$. (*Hint:* Exercise 3.2.18.)

(b) Let p be prime. Show that there is a primitive root modulo p, i.e. a number a with $\text{ord}_p(a) = p - 1$.

(*Hint:* If $\text{ord}_p(a) < p-1$, then a is a root (modulo p) of some polynomial K_r where $r \mid (p-1)$. What is the largest number of roots this polynomial can possibly have?)

6.4.10. The cyclotomic polynomial K_r is irreducible over $\mathbb{Z}$, even if r is composite. However, this is more difficult to prove than in the case where r is prime.

6.4.11. Exercise. This exercise is for readers who are familiar with the **complex numbers**; see also Comment A.21. A complex number z is called an **r-th root of unity** if $z^r = 1$. So the r-th roots of unity are exactly the complex roots of the polynomial $X^r - 1$. Furthermore z is a **primitive r-th root of unity** if also $z^m \neq 1$ for every number m with $1 \leq m < r$.

(a) Where are the roots of unity situated (geometrically) in the complex number plane?

(b) Prove that there are exactly $\varphi(r)$ primitive r-th roots of unity. (*Hint:* The r-th roots of unity are precisely the numbers $e^{2\pi i q/r}$, where $q \in \{0, \ldots, r-1\}$.)

(c) Show by induction that the r-th cyclotomic polynomial is given by

$$K_r(X) = \prod_{\substack{z \text{ primitive } r\text{-th} \\ \text{root of unity}}} (X - z). \tag{6.4.12}$$

This equation gives an alternative definition of cyclotomic polynomials. Indeed, if we define K_r by (6.4.12), then these polynomials automatically satisfy (6.4.7). It remains to show that all coefficients are integers. (From the definition we only know that they are complex numbers.) It is not too difficult to do so by induction and we invite the reader to work out the details.

6.4.13. Concluding this chapter, we remark that we could also study the irreducible factors of K_r modulo a prime number p, where we assume that r and p are coprime. In this setting, Lemma 6.4.2 still holds, but the proof becomes more difficult.

Recall that this lemma was the only place in the proof of Theorem 6.1.1 at which the primality of r was used. Hence this assumption could actually be removed from the statement of the theorem.

Chapter 7

The algorithm

Having proved the theorem of Agrawal, Kayal, and Saxena, we can now formulate the AKS algorithm and prove that it is indeed a deterministic and efficient algorithm for the problem PRIMALITY. We first need to answer the problem of choosing the polynomial Q (i.e. how to choose the number r from Theorem 6.1.1), which had been left open so far. This is explained in the first section of this chapter; after this we develop the final form of the algorithm. We end with a short discussion of possible improvements and new developments.

7.1. How quickly does the order of n modulo r grow?

By Theorem 6.1.1, we know that we can detect a composite number by testing only a small number of congruences – provided that there is a suitable prime number r that is not too large itself! Recall that r should satisfy

$$\mathrm{ord}_r(n) > 4(\log n)^2$$

and that it should grow at most polynomially with $\log n$. Is there always such a number r? In order to reformulate this question we introduce some more notation.

7.1.1. Definition.
Let $n \geq 2$ and $k \in \mathbb{N}$. We use $r(n, k)$ to denote the smallest prime r such that $r \mid n$ or $\operatorname{ord}_r(n) > k$.

Our question is: how large is $r(n, k)$, depending on n and k? We give an elementary answer that is quite rough, but sufficient for our purposes. Interestingly, the search for better bounds quickly leads to deep theorems and conjectures from number theory; see Section 7.3.

Recall that if $\operatorname{ord}_p(n) = m$, then $n^m \equiv 1 \pmod{p}$ and therefore $n^m - 1$ is a multiple of p. Thus the number

$$N := \prod_{m \leq k} (n^m - 1)$$

is a common multiple of all prime numbers p that satisfy $\operatorname{ord}_p(n) \leq k$. In particular, N is a common multiple of all primes $p < r(n, k)$.

The smallest common multiple of all these primes is their product

$$\Pi := \prod_{\substack{p \text{ prime},\\ p < r(n,k)}} p, \tag{7.1.2}$$

so we have $\Pi \leq N$. The number N depends on n and k, and we can estimate Π from below, for example by using the weak prime number theorem. In this way, we obtain the desired upper bound for $r(n, k)$.

7.1.3. Theorem (Size of $r(n, k)$)**.**
The number $r(n, k)$ satisfies

$$r(n, k) = O\big(k^4 \cdot (\log n)^2\big).$$

(I.e. there is a number K such that $r(n, k) \leq K \cdot k^4 \cdot (\log n)^2$ for all n and k.)

Proof. We simplify notation by writing r instead of $r(n, k)$. The weak prime number theorem (Theorem 4.3.3) tells us that Π grows at least exponentially in $r/\log r$. More precisely,

$$\Pi \geq 2^{\pi(r-1)} \geq 2^{\frac{C\cdot(r-1)}{\log(r-1)}} > 2^{\frac{C\cdot(r-1)}{\log r}} > 2^{\frac{C\cdot r}{2\log r}}.$$

(Here Π is the number from (7.1.2), $\pi(r-1)$ is, as in Chapter 4, the number of primes $p < r$, and C is a suitable constant that is independent of n, k, and r.)

But we also know that N grows at most exponentially in $k^2 \cdot \log n$, as we can see from

$$N = \prod_{m \leq k} (n^m - 1) < \prod_{m \leq k} n^m = n^{1+\cdots+k} = n^{\frac{k(k-1)}{2}} < n^{\frac{k^2}{2}} = 2^{\frac{k^2 \cdot \log n}{2}}.$$

(Here we used the Gauss summation formula from Exercise 1.1.12.) Comparing these two estimates and remembering that $\Pi \leq N$, we obtain

$$\frac{r}{\log r} < \frac{k^2 \cdot \log n}{C}.$$

Now that we have an estimate for $\frac{r}{\log r}$, we can easily deduce one for r as well. Indeed, we know that $(\log(r))^2 = O(r)$. (See Exercise 2.3.4.) So in particular $r \cdot (\log(r))^2 = O(r^2)$ and thus

$$r = O\left(\frac{r^2}{\log(r)^2}\right) = O(k^4 \cdot (\log n)^2). \qquad \blacksquare$$

For the application in Theorem 6.1.1, we are interested in the order of magnitude of $r_0 := r(n, \lfloor 4(\log n)^2 \rfloor)$. By Theorem 7.1.3, this number satisfies

$$r(n, \lfloor 4(\log n)^2 \rfloor) = O((\log n)^{10}). \tag{7.1.4}$$

Thus r_0 grows at most polynomially with $\log n$, as desired. So there is no problem in finding this number and hence the corresponding polynomial $Q = X^{r_0} - 1$ efficiently, using a simple search. Thus we have solved point (2) from our strategy in Section 5.2, which was the final problem remaining in our development of the algorithm.

Exercises.

7.1.5. Exercise. Show that, for every $\varepsilon > 0$, even

$$r(n,k) = O\left((k^2 \cdot \log n)^{1+\varepsilon}\right)$$

holds in Theorem 7.1.3. Deduce that, in (7.1.4), we could replace $(\log(n))^{10}$ by $(\log(n))^{5+\varepsilon}$.

7.2. The algorithm of Agrawal, Kayal, and Saxena

Here we close the gaps in the basic structure of the AKS algorithm from Section 5.2. This establishes the algorithm in its final form:

```
ALGORITHM AKS
Input: A natural number $n \geq 2$.
  1. If $n$ is a perfect power, i.e. $n = a^b$ with $a < n$ and
     $b > 1$, then answer "$n$ is composite".
  2. Otherwise do the following for $r = 2, 3, 4, \ldots$:
       (a) Test whether $r$ is prime (e.g. by using the
           Sieve of Eratosthenes).
       (b) If $r \mid n$ and $r < n$, answer "$n$ is composite".
       (c) If $r \geq n$, answer "$n$ is prime".
       (d) Otherwise compute the order $\mathrm{ord}_r(n)$.
       (e) If $r$ is prime and $\mathrm{ord}_r(n) > 4(\log n)^2$, set
           $Q := X^r - 1$ and continue with Step 3.
  3. Test the congruences
         $(X + a)^n \equiv X^n + a \pmod{n, Q}$
     for all integers $a$ from 1 to $r - 1$.
  4. If one of these congruences is not satisfied, answer
     "$n$ is composite".
  5. Otherwise answer "$n$ is prime".
```

7.2.1. Theorem.
Algorithm AKS is deterministic and efficient. If n is a prime number, then the output of the algorithm is "n is prime". Otherwise, the algorithm outputs "n is composite".

Proof. We have effectively proved the theorem already, but to be safe we shall explicitly check all claims one by one.

Clearly the algorithm is deterministic. We must check that each step can be carried out efficiently. For Step **1**, the recognition of perfect powers, we saw in Exercise 2.3.6 that this is the case.

Step **2** is repeated until we have found the number

$$r_0 := r(n, \lfloor 4(\log n)^2 \rfloor)$$

from the previous section. By (7.1.4), we know that r_0 grows at most polynomially in $\log n$; so in particular the number of times this step is repeated is at most polynomial in $\log n$. The time required to check whether r is prime is polynomial in r, and testing whether n and r are coprime requires only an application of the Euclidean algorithm, which is efficient. The same is true for computing the order $\operatorname{ord}_r(n)$ because this requires at most r multiplications modulo r. In total, the running time of the second step is at most polynomial in $\log n$.

The number of congruences that are tested in Step **3** is at most r_0-1, hence again polynomial in $\log n$. We saw in Exercise 5.1.10 how each of these congruences can be checked efficiently. (This was one of the key ideas in coming up with the algorithm.) Thus the running time of this step is bounded by a polynomial function of $\log n$.

So the algorithm is efficient; we must check correctness. If the algorithm answers "n is composite", one of the following happened:

- n was recognized as a perfect power of some number $a < n$ in the first step.
- In Step **2**(b), the algorithm found a prime divisor r of n such that $r < n$.
- One of the congruences in the third step is not satisfied. Then n is composite by Theorem 5.1.1.

So in each of these cases, n is indeed a composite number.

If, on the other hand, there are two cases where the algorithm answers "n is prime":

- In Step **2**(c), we have $r \geq n$. Then we have tested all numbers $r < n$ in Step **2** without finding a prime divisor of n (otherwise the algorithm would have stopped, answering "n is composite"). Thus n is prime.
- All congruences in Step **3** are satisfied, and by Theorem 6.1.1, n is either a prime or a prime power. In the latter case, the algorithm would have already stopped in Step **1**, so n must be prime.

In summary, the algorithm answers "n is prime" *if and only if* n is prime. Thus the theorem has been proved, and we have achieved the goal that we set for ourselves at the beginning of the book. ■

Exercises.

7.2.2. Exercise (P). Implement the AKS algorithm. (Naturally this will require using many operations from earlier programming exercises.)

Further Exercises and Comments.

7.2.3. Using Comment 6.1.4 we could improve the algorithm a little bit. First of all, we can replace the bound of $r-1$ in the third step by the smaller number $2\sqrt{r}\log n$, which speeds up the execution.

Furthermore, it is not necessary to require the primality of the number r. Again, this will speed up the execution because we can omit the primality test on r and because the smallest natural number r with $\operatorname{ord}_r(n) > 4(\log n)^2$ may not be prime.

7.2.4. Exercise. We developed the AKS algorithm on the basis of approach (b) from Section 5.2. Here we will see that Theorem 6.1.1 can also be used to obtain a primality test on the basis of approach (a). That is, we will find an algorithm that tests a *fixed* congruence modulo several *different* polynomials Q. (This observation comes from the article of Agrawal and Biswas [**AB**].)

(a) Let $n \geq 2$ and let P, Q, and T be polynomials. Show: if Q is a divisor of P modulo n, then $Q(T)$ is a divisor of $P(T)$ modulo n.

(b) Deduce that if P_1, P_2, and Q are polynomials and $a \in \mathbb{Z}$, then $P_1 \equiv P_2 \pmod{n, Q}$ if and only if

$$P_1(X+a) \equiv P_2(X+a) \pmod{n, Q(X+a)}.$$

(c) Let $n \geq 2$ and $r \in \mathbb{N}$. Use induction to prove the following for all $\ell \in \mathbb{N}$: if

$$(X-1)^n \equiv X^n - 1 \pmod{n, (X-a)^r - 1} \tag{7.2.5}$$

for all $a = 1, \ldots, \ell$, then also

$$(X+a)^n \equiv X^n + a \pmod{n, X^r - 1}$$

for all $a = 1, \ldots, \ell$.

This means that, in Step **3** of the AKS algorithm, we could just as well test the congruence (7.2.5) for all numbers a from 1 to $r-1$.

7.3. Further comments

As we have already emphasized several times, the goal of our treatment of the AKS algorithm was to give a complete and accessible proof. In particular, our presentation differs somewhat from that in the original paper by Agrawal, Kayal, and Saxena, which placed more emphasis on obtaining a better running time.

We have already mentioned some possibilities for improvement in exercises and comments at the end of the relevant sections. Others can be obtained by using the best known algorithms for arithmetic with polynomials. Information on these can be found in the literature on efficient algorithms. We now briefly mention some possible improvements in our estimates for the running time of the algorithm.

The size of r_0. If we look at the algorithm carefully, we notice that it is Step **3** that takes the longest amount of time. How long this is exactly depends on how large the number r_0 is. So we can only hope for clear improvements in our estimates of the running time if we can find better upper bounds for the number $r_0 = r(n, \lfloor 4(\log n)^2 \rfloor)$, which was defined as the smallest prime r_0 with $\mathrm{ord}_{r_0}(n) > 4(\log n)^2$. Our original bound in (7.1.4) had the order of magnitude $O\left((\log n)^{10}\right)$. By Exercise 7.1.5 this can be improved to $O\left((\log n)^{5+\varepsilon}\right)$, where ε is an arbitrarily small positive number. If we do not require that r_0 be prime (Comment 7.2.3), then we can improve this bound a little bit further: in this case we obtain $r_0 = O((\log n)^5)$ (Exercise 7.3.2).

It is possible to strengthen these bounds further, but as already mentioned in Section 7.1, this leads to some very deep theorems and questions of number theory whose surface we can only scratch in this book. (For more details we refer the reader to the original paper as well as the literature mentioned at the end of the chapter.)

From now on, we fix n and are still looking for a number r for which $\mathrm{ord}_r(n)$ is "big". By the Fermat-Euler Theorem, we know that $\mathrm{ord}_r(n)$ is a divisor of $\varphi(r)$. So it would be useful to find numbers for which $\varphi(r)$ itself has big prime divisors.

The study of prime numbers r for which $\varphi(r) = r - 1$ has large prime divisors has a long history: it goes back almost two centuries, to the work of the French mathematician Sophie Germain. While working on Fermat's Last Theorem, she studied prime numbers p for which $k \cdot p + 1$ is also prime, where k is an even number. An important special case is given by $k = 2$.

7.3.1. Definition (Sophie Germain prime).
A prime p is called a *Sophie Germain prime* if $2p + 1$ is also prime.

In analogy to the prime counting function $\pi(m)$ we can denote by $S(m)$ the number of Sophie Germain primes p with $p \leq m$. In 1922, the English mathematicians Hardy and Littlewood formulated the following conjecture: *there is a constant C such that $S(M)$ behaves asymptotically like $Cm/(\log m)^2$ as $m \to \infty$.*

If this is right, then it follows that the order of magnitude of our number r_0 is at most $O((\log n)^{2+\varepsilon})$, where $\varepsilon > 0$ can be chosen arbitrarily small. (See Exercise 7.3.4.) So our algorithm would find a suitable number r much faster than our previous bounds suggest. Unfortunately the conjecture on Sophie Germain primes is still unproven – as discussed in Appendix A, it is not even known whether there are infinitely many such primes!

On the other hand, there is a theorem of Fouvry from 1985 that studies prime numbers q that, while perhaps not being Sophie Germain primes, at least have the property that $q - 1$ has a prime factor that is much larger than $\sqrt{q}$. Using this deep and extremely difficult result from analytic number theory, one can show at least that $r = O((\log n)^3)$, which is still a considerable improvement in our original bound.

Variants of the AKS algorithm. Soon after Agrawal, Kayal, and Saxena published their article, a number of people have developed improvements. Here we only want to mention an algorithm of Lenstra and Pomerance [**LP**], which has a better running time than the AKS algorithm. The basic idea is, essentially, the same as that of Agrawal, Kayal, and Saxena, but a different polynomial Q is used to test the congruences. However, here the proof is considerably more difficult, and we are unable to give further details here.

Despite all improvements in running time, there is still no known deterministic primality test that can remotely hope to compete with that of Miller and Rabin. Therefore, there is still much room for exciting new developments!

Exercises.

7.3.2. Exercise. Let $r = r'(n,k)$ be the smallest natural number satisfying $\gcd(n,r) \neq 1$ or $\operatorname{ord}_r(n) > k$. Show that $r'(n,k) = O(k^2 \log n)$.

Hint: Follow the same idea as in Section 7.1, but use Theorem 4.4.3 instead of the weak prime number theorem.

Further Exercises and Comments.

7.3.3. Sophie Germain was an influential French mathematician in the early nineteenth century. The scope and importance of her work, in particular on Fermat's Last Theorem, was long underestimated; see [**LaPe**].

7.3.4. Exercise.

(a) Let p be a Sophie Germain prime and set $r := 2p+1$. Show: for every natural number n that is not a multiple of r, we have $\operatorname{ord}_r(n) \leq 2$ or $\operatorname{ord}_r(n) \geq p$.

(b) Show: for every number $n \geq 2$ there are at most $2\log n$ prime numbers r satisfying $\operatorname{ord}_r(n) \leq 2$.

(c) Assume that the conjecture of Hardy and Littlewood on Sophie Germain primes, as stated in the preceding section, is correct.

Deduce: if $\varepsilon > 0$ is any positive number and if $k \in \mathbb{N}$ is sufficiently large, then there are at least k Sophie Germain primes p satisfying $k < p \leq k^{1+\varepsilon}$.

(d) Deduce (still under the assumption that the conjecture is correct) that for every $\varepsilon > 0$, we have

$$r(n,k) = O((\max(\log n, k))^{1+\varepsilon}).$$

(Here $\max(\log n, k)$ denotes the larger of the two numbers $\log n$ and k.)

7.3.5. If r is prime, let us denote by $P(r)$ the largest prime divisor of $r-1$. The theorem of Fouvry from 1985 mentioned above states the following:

There is a constant $\delta > 2/3$ and a constant $c_\delta > 0$ such that for every $m \geq 2$ there are at least $c_\delta \cdot m/\log m$ prime numbers r satisfying $P(r) > r^\delta$.

This theorem also has a connection to Fermat's Last Theorem: with its help, Fouvry was able to show in joint work with Adleman and Heath-Brown, also in 1985, that the "first case" of Fermat's Last Theorem is satisfied at least for infinitely many prime numbers. (Fermat's Last Theorem was completely proved by Andrew Wiles roughly ten years later.)

7.3.6. The proof of Fouvry's Theorem, like other results from analytic number theory, has an interesting property. The constant c_δ is "not effective": it is proved that this number exists, but there is no $\delta > 1/2$ for which any explicit choice c_δ is known.

The reason for this is that, in the end, the proof relies on distinguishing two cases. First it is shown that the theorem is correct if the Generalized Riemann Hypothesis is true. (In this case, it is possible to make explicit statements about the constant c_δ.) On the other hand, it is shown that the theorem is also correct if this conjecture is wrong, and then the constant depends on the smallest counterexample! (See [**G**, Section 5].) This is an impressive example of the power of indirect mathematical proofs.

Further reading

The article [**Bo**] contains an interesting summary of the AKS algorithm and its development. The book [**Dtz**] and the survey article [**G**] are also recommended for those who would like to learn more about the algorithm and related results. Finally, we again mention the book [**CP**], which contains an impressive variety of results about primality tests, factorization methods, and much more. It is highly recommended to more advanced readers.

Appendix A

Open questions

In this appendix, we will present some more results concerning prime numbers and discuss some problems that remain unsolved. We shall not give proofs, and of course our discussion of open questions is not complete but is rather intended as a starting point for those readers who are interested in problems related to prime numbers. There will also be plenty of suggestions for further reading.

The Riemann Hypothesis. The Riemann Hypothesis is undoubtedly one of the most famous open problems of modern mathematics. We have already encountered it in Comment 4.3.6.

A.1. Conjecture (Riemann Hypothesis as phrased by von Koch)**.** Let $\pi(n)$ denote the prime number function and let $\mathrm{Li}(n) = \int_2^n \frac{\mathrm{d}t}{\ln t}$ denote the Logarithmic Integral Function. Then

$$\big|\pi(n) - \mathrm{Li}(n)\big| = O(\sqrt{n}\ln n).$$

The usual version of the Riemann Hypothesis is harder to explain because it concerns the so-called **Riemann ζ-function**. This function is defined on the set $\mathbb{C}$ of *complex numbers* (see Comment A.21). For elements $s \in \mathbb{R}$ such that $s > 1$, the value of the ζ-function can be expressed in terms of a series in the following form:

$$\zeta(s) := \sum_{n=1}^{\infty} \frac{1}{n^s}.$$

Euler noticed that

$$\zeta(s) = \prod_p \frac{1}{1 - p^{-s}}$$

where the product is taken over all prime numbers p. Hence we see that there is a connection between the ζ-function and the set of prime numbers. (For a proof of this statement and related results we refer the reader to [**HW**] and [**J**].)

In a ground-breaking article from 1859, Riemann proved that the ζ-function can be extended to a function that is defined on the entire complex plane, except at the point 1, where it has a pole (i.e. takes the value ∞). It is also known that all negative even numbers are zeros of the ζ-function. In his paper mentioned above, Riemann stated the following conjecture about all complex zeros of the extended function:

A.2. Conjecture (Riemann Hypothesis)**.**
Suppose that $z = x + iy$ is a zero of the ζ-function and that $y \neq 0$. Then $x = \frac{1}{2}$.

Geometrically, this would mean that all non-real zeros of the ζ-function lie on a vertical line (in the complex plane) going through $\frac{1}{2}$, called the **critical line**. It is well known that all non-real zeros must belong to the **critical strip** consisting of numbers $z = x + iy$ with $0 < x < 1$.

Von Koch proved in 1901 that the original form of the Riemann Hypothesis is equivalent to Conjecture A.1. The ζ-function also plays a role in the usual proof of the prime number theorem, Theorem 4.3.2.

In 1900, Hilbert gave a list of 23 deep mathematical problems that where central to mathematics in his opinion, and the Riemann Hypothesis was one of these. It is still neither proved nor disproved, and as one of the "Millennium Prize Problems" of the Clay Institute, its proof would be rewarded with a million US dollars. (The problem $\mathbf{P} \overset{?}{=} \mathbf{NP}$ from Section 2.4 is also on this list.)

There is a lot of research going on around the Riemann Hypothesis. It is widely believed to be true (sometimes it is used as an explicit additional assumption in the proof of certain mathematical results) and computer calculations show that at least the first 10 000 000 000 000 complex, non-real zeros of ζ satisfy the conjecture. Therefore the discovery of a counterexample would be quite a surprise (but to our knowledge there would be no financial reward in this case).

We would also like to mention the so-called **Generalized Riemann Hypothesis**; see also Comment 4.5.11. It says that each member of a certain class of functions, including the Riemann ζ-function, satisfies Conjecture A.2.

The Goldbach Conjecture. In 1742 Goldbach wrote, in a letter to Euler:

> *It appears at least that every number that is greater than* 2 *is a sum of three prime numbers.*

We should mention here that Goldbach considered 1 to be a prime number, contrary to our convention. Hence a modern version of Goldbach's statement would be that all numbers that are at least 6 can be written as a sum of three prime numbers. Examples are $8 = 3+3+2$ and $13 = 5+5+3$. When answering Goldbach, Euler mentioned that this can be rephrased in the following way:

A.3. Conjecture (Goldbach Conjecture)**.**
Every even number $n \geq 4$ can be written as a sum of two prime numbers.

This is the most well-known version of the Goldbach Conjecture; see also Exercise A.22. In spite of many attempts by professional mathematicians as well as interested amateurs, and in spite of the beautiful simplicity of the statement, a proof does not appear to be even remotely in reach. Some progress on the conjecture was made by Chen in the 1970s; as far as we know this remains the most recent significant contribution to this problem in a long time. A simplified proof of Chen's result is contained in [**Ros**].

A.4. Theorem (Chen, 1973)**.**
If n is a sufficiently large *even number, then n can be written as $n = p_1 + p_2$ with a prime number p_1 and a number p_2 that is a product of at most two primes.*

Much more is known as soon as we replace "two" by a larger even number in Goldbach's statement. For example, in 1930 it was proved that all numbers $n \geq 4$ can be written as a sum of at most 300 000 prime numbers. In 1995, Ramaré showed that at most six prime numbers are enough (quite an improvement from 300 000!). As far as we know, this was the best known result of this type prior to Helfgott's work from 2013 (see below).

The following is a famous special case of the conjecture for odd numbers:

A.5. Conjecture (Weak Goldbach Conjecture)**.**
All natural numbers $n > 7$ can be written as a sum of three odd prime numbers.

Already in 1937, Vinogradov was able to prove the following result:

A.6. Theorem (Vinogradov's Theorem)**.**
If $n \in \mathbb{N}$ is a sufficiently large odd natural number, then n can be written as a sum of three odd primes.

This meant that, in order to establish the Weak Goldbach Conjecture, it would, in principle, suffice to check only finitely many numbers – but how many? Vinogradov's original proof does not provide an actual bound on this number. Although subsequent mathematicians were able to rectify this, until very recently the best known versions of Vinogradov's Theorem established the Weak Goldbach Conjecture for numbers n greater than 10^{1350}. This is far larger than the estimated number of atoms in the universe, so there was no hope of checking this many numbers computationally.

However, in 2013 the Peruvian mathematician Harald Helfgott announced a complete proof of the Weak Goldbach Conjecture [**He**]. More precisely, he was able to improve on Vinogradov's Theorem by proving that the conjecture holds for all $n \geq 10^{29}$; the remaining cases can then be ruled out using some (quite sophisticated) numerical verification of the Goldbach Conjecture and the Riemann Hypothesis within a certain range of numbers. We note that it had already been proved in 1997 that the Weak Goldbach Conjecture is true if the Generalized Riemann Hypothesis holds.

Just as Fermat's Last Theorem, the Goldbach Conjecture is a typical example of mathematical problems that are easy to state (as often is the case in number theory) but turn out to be extremely difficult. The mathematics that is developed when trying to solve such a problem often has much greater impact than the result itself, leading to new methods and concepts and hence advancing all relevant research areas.

Remark. The Goldbach Conjecture should not be confused with the statement that all prime number $p \geq 3$ can be written as a difference of two squares. This is true and in fact not difficult to prove (see Exercise A.23). Another result worth mentioning in this context is once again due to Fermat: he proved that all prime numbers that are congruent to 1 modulo 4 can be written in a unique way as a sum of two squares.

Twin Primes. For all $n \in \mathbb{N}$, the pair $(n, n+2)$ is called a pair of **twin primes** if and only if n and $n+2$ are prime. We have already mentioned such pairs in Comment 1.6.5; some examples are $(3, 5)$, $(5, 7)$, and $(1997, 1999)$. The largest pair of currently known twin primes (as of 2013) consists of two numbers with 200 700 digits.

A.7. Conjecture.
There are infinitely many twin primes.

We do not know the origin of this conjecture or who explicitly stated it for the first time, but in the first edition of [**HW**] there is already a stronger version:

A.8. Conjecture.
There are infinitely many triples of prime numbers $(n, n+2, n+6)$ and $(n, n+4, n+6)$.

(On the other hand, the reader is invited to verify in Exercise A.24 that $(3, 5, 7)$ is the only prime triple of the form $(n, n+2, n+4)$.)

Just as for the Goldbach Conjecture, Chen was able to prove a version of the conjecture where the requirement that both numbers be primes is weakened:

A.9. Theorem (Chen, 1973)**.**
For all even numbers $h \in \mathbb{N}$ there are infinitely many pairs $(p, p+h)$ such that p is prime and $p+h$ is a product of at most two prime numbers.

Note that the Twin Primes Conjecture concerns the size of *prime gaps*, i.e. the distance between one prime number and the next one. In Theorem 1.6.2, we proved that there are arbitrarily large prime gaps, and the prime number theorem implies that the *average* gap between two primes becomes larger as the numbers increase. With this terminology, the conjecture claims that, nonetheless, there are infinitely many gaps of size two. In April 2013, Zhang [**Zh**] announced a proof of the following theorem.

A.10. Theorem (Bounded gaps between primes)**.**
There is some $H \in \mathbb{N}$ such that there are infinitely many prime gaps of size at most H.

In other words, there are infinitely many pairs (p, q) of primes such that $p < q$ and $q - p \leq H$.

The result was immediately hailed as a major breakthrough and was accepted for publication in the *Annals of Mathematics*, one of the most prestigious mathematical journals, within one month. How large is the number H? Zhang's original article gives the rather large number $H = 70\,000\,000$. However, his work prompted an intensive international online collaboration between mathematicians, coordinated by Terence Tao, to improve the bounds occurring in Zhang's work. As of October 2013, they have succeeded in reducing the value of H to less than 5000. Nonetheless, it appears that new ideas would be required to obtain $H = 2$ and hence prove the Twin Primes Conjecture.

Finally, let us note that alternative characterizations of twin primes have been known for a while, for example the following one due to Clement (see also Exercise A.25):

A.11. Theorem (Clement's Theorem, 1949)**.**
Let $n \in \mathbb{N}$. Then $(n, n+2)$ is a pair of twin primes if and only if $4((n-1)!+1)+n$ is divisible by $n(n+2)$.

As mentioned above, there is only one triple of prime numbers of the form $(n, n+2, n+4)$. However, it is easy to find triples of prime numbers of the form $(n, n+3, n+6)$ or quadruples of prime numbers of the form $(n, n+6, n+12, n+18)$, for example $(3, 7, 11)$ and $(5, 11, 17, 23)$, respectively. Generally speaking, if $x, c, N \in \mathbb{N}$, then a sequence of the form $x, x+c, x+2c, \ldots, x+Nc$ is a **(finite) arithmetic progression**. The number N of elements in such a sequence is called the **length** of the progression.

For a long time, it was an open question whether or not it is possible to find arithmetic progressions consisting only of prime numbers and of arbitrary length. This problem was finally solved by Ben Green and Terence Tao in 2004; their remarkable article appeared in the *Annals of Mathematics* in 2008 [**GT**].

A.12. Theorem (Green-Tao Theorem)**.**
The set of prime numbers contains finite arithmetic progressions of arbitrary length.

Dirichlet's Prime Number Theorem can be viewed as a converse to the Green-Tao result. Its proof is quite difficult (published in 1837 in the *Abhandlungen der Königlichen Preußischen Akademie der Wissenschaften*) and is therefore often omitted in number theory books such as [**HW**].

A.13. Theorem (Dirichlet's Prime Number Theorem).
Let $a \in \mathbb{N}$ and let $n \in \mathbb{N}$ be coprime to a. Then the (infinite) arithmetic progression $a, a+n, a+2n, a+3n, \ldots$ contains infinitely many prime numbers.

Mersenne numbers. For any $n \in \mathbb{N}$, the n-th **Mersenne number** M_n is defined as

$$M_n := 2^n - 1.$$

Mersenne numbers naturally appear in the context of so-called **perfect numbers**. A number $n \in \mathbb{N}$ is called **perfect** if and only if n is the sum of all divisors of n that are strictly smaller than n. For example, 6 and 28 are perfect numbers because $6 = 1 + 2 + 3$ and $28 = 1 + 2 + 4 + 7 + 14$. Euclid proved the following:

A.14. Theorem (Euclid).
Suppose that $n \in \mathbb{N}$ is such that $2^n - 1$ is prime. Then $2^{n-1}(2^n - 1)$ is a perfect number.

It was long conjectured that the numbers from Theorem A.14 in fact are the only even perfect numbers. The first known proof is due to Euler.

A.15. Theorem (Euler).
A number $a \in \mathbb{N}$ is perfect if and only if there exists some $n \in \mathbb{N}$ such that $a = 2^{n-1}(2^n - 1)$ and $2^n - 1$ is prime.

For this reason, Hardy and Wright [**HW**, Section 16.8] refor to the even perfect numbers as "Euclid numbers". (It is not known whether or not there are any odd perfect numbers.) The results of Euclid and Euler reveal a connection between perfect numbers and

Mersenne numbers, specifically Mersenne numbers that are prime. Hence we define

A.16. Definition (Mersenne prime)**.**
A **Mersenne prime** is a Mersenne number that is prime.

Hence $M_2 = 2^2 - 1 = 3$ and $M_3 = 2^3 - 1 = 7$ are Mersenne primes, but $M_4 = 2^4 - 1 = 15$ is not. It is an exercise that if $n \in \mathbb{N}$ is composite, then M_n is composite too (Exercise A.28). This raises a number of questions – in fact, the investigation of Mersenne numbers is a very active research area, which has led to the discovery of many interesting number-theoretic results. Here we will just discuss the following questions:

- Is it true that for all primes p the p-th Mersenne number is prime?
- How many Mersenne primes are there?

The answer to the first question is *No*, and the first counterexample is given already by $M_{11} = 2047 = 23 \cdot 89$. So we need an additional hypothesis, leading to a primality test for Mersenne numbers.

A.17. Theorem (Lucas test)**.**
We define a sequence of natural numbers $k_0, k_1, \ldots$ recursively by setting $k_0 := 4$ and $k_{i+1} := k_i^2 - 2$ for all $i \geq 0$.

Now suppose that $n \geq 3$ is prime. Then M_n is a Mersenne prime if and only if M_n divides k_{n-2}.

This result can be proved within a few pages once a sufficient number-theoretic background is established. The second question we asked above is still wide open: it is conjectured but not known that there are infinitely many Mersenne primes. The search for new Mersenne primes is being pursued quite vigorously by a distributed computing project known as the "Great Internet Mersenne Prime Search". As of the time of this writing, the largest known Mersenne prime is $M_{57\,885\,161}$, which was discovered in January 2013. (The previous record holder, $M_{43112609}$, had been found in 2008.) This is also the largest known prime number. In 1964–1976, the post office at the University of Illinois mathematics department even used a cancellation stamp dedicated to the Mersenne prime M_{11213} (which had just been discovered there)!

Sophie Germain primes. Sophie Germain primes were already mentioned in Chapter 7: a prime $p \in \mathbb{N}$ is called a **Sophie Germain prime** if $2p+1$ is prime as well. The numbers 2, 3, 5, and 11 are examples. The largest currently known Sophie Germain prime is $18\,543\,637\,900\,515 \times 2^{666\,667} - 1$ and was found in April 2012.

A.18. Conjecture.
There are infinitely many Sophie Germain primes.

No proof of this conjecture is known yet, but occasionally new Sophie Germain primes are found. These numbers have a connection to Mersenne primes and to Fermat's Last Theorem (which Sophie Germain had been working on). Sophie Germain herself was in fact more generally interested in prime numbers p that have the property that, for some even $n \in \mathbb{N}$, the number $np+1$ is also prime. Around 1825 she proved that Fermat's Last Theorem holds for exponents that are prime with this special property and that satisfy some technical conditions. For $n = 2$ these technical conditions turn out to be automatically satisfied, so Fermat's Last Theorem was known to be true for all such exponents, which were therefore later named after Sophie Germain.

If $p > 3$ is a Sophie Germain prime such that $p \equiv 3 \pmod 4$, it is a – slightly tricky – exercise to show that the Mersenne number M_p is composite. Even more is true [**HW**, Theorem 103]:

A.19. Theorem.
Suppose that p is a prime number, that $p > 3$, and that $p \equiv 3 \pmod 4$. Then p is a Sophie Germain prime if and only if the p-th Mersenne number M_p is divisible by $2p+1$.

The exception $p = 3$ is necessary since $M_3 = 2^3 - 1 = 7 = 2 \cdot 3 + 1$. This means that $2p+1$ divides M_p, but this time M_p is not composite.

Fermat numbers. For all $n \in \mathbb{N}$ the n-th Fermat number is defined to be $F_n := 2^{2^n} + 1$. A Fermat number is called **Fermat prime** if and only if it is prime.

Fermat himself conjectured in 1650 that all numbers of the form $2^{2^n} + 1$ are prime. But this quickly turned out to be wrong: for example, F_5 is composite. It remains an open question whether or not there are infinitely many Fermat primes.

A.20. Conjecture (Eisenstein, 1844)**.**
There are infinitely many Fermat primes.

So far it is known that F_0, F_1, F_2, F_3, F_4 are prime but that all larger Fermat numbers up to F_{32} are composite. It is very difficult to factorize numbers of this size, which is why full factorizations currently are known only up to F_{11}. Moreover, F_{20} and F_{24} are known to be composite, but *no prime factor* has been found so far! We are not aware of any theoretical results that bring us closer to understanding how many Fermat primes there might be.

Further Exercises and Comments.

A.21. *Complex numbers* are written in the form $z = x + iy$, where i is an "imaginary unit" satisfying $i^2 = -1$. They can be added, multiplied, and divided using the usual rules of arithmetic; for example

$$(1+i)\cdot(1-i) = 1 - i^2 = 2 \qquad \text{and}$$
$$\frac{1+i}{1-i} = \frac{(1+i)^2}{(1+i)(1-i)} = \frac{1+2i+i^2}{2} = i.$$

Indeed, the set of complex numbers, denoted by $\mathbb{C}$, is a *field* in the sense of Comment 3.1.24. Just as we imagine the real numbers as forming a line, we can visualize the complex numbers as filling a plane, where x and y represent the horizontal, resp. vertical, coordinates.

Although this construction may at first seem counterintuitive and artificial, it turns out that complex numbers are extremely natural, and their study can provide deep insight into problems that, at first glance, are firmly rooted in the world of real numbers. The connection between primes and the Riemann ζ-function is one example of this.

A.22. Exercise. Prove: all $n \in \mathbb{N}$ with $n \geq 6$ can be written as a sum of (exactly) three prime numbers if and only if all even numbers $n \geq 4$ can be written as a sum of (exactly) two prime numbers.

A.23. Exercise. Prove that all odd numbers $m \geq 3$ can be written as a difference of two perfect squares. (*Hint:* Write $m = 2k+1$ such that $k \in \mathbb{N}$. How is this related to $(k+1)^2$?)

A.24. Exercise. Prove that $(3, 5, 7)$ is the unique triple of prime numbers of the form $(n, n+2, n+4)$. (*Hint:* Exercise 1.2.8).

Moreover, prove that all pairs of twin primes except $(3, 5)$ can be written as $n = 6k - 1$ and $n + 2 = 6k + 1$ with suitable $k \in \mathbb{N}$.

A.25. Exercise. Recall Wilson's Theorem from Exercise 4.5.10: a number $n > 1$ is prime if and only if $(n-1)! + 1 \equiv 0 \pmod{n}$. Use this result to prove Clement's Theorem as follows:

(a) Let $n \in \mathbb{N}$. Prove that $n + 2$ is prime if and only if

$$2(n-1)! + 1 \equiv 0 \pmod{n+2}.$$

(b) Let $n \in \mathbb{N}$ be such that n and $n + 2$ are both prime. Prove that

$$4\big((n-1)! + 1\big) + n \equiv 0 \pmod{n(n+2)}. \tag{A.26}$$

(c) Conversely suppose that $n > 1$ is odd and that (A.26) holds. Deduce that n and $n + 2$ are prime.

(d) Prove that if n is even, then $2 \cdot \big(2(n-1)! + 1\big) \not\equiv 0 \pmod{n+2}$. Deduce that (A.26) never holds for odd numbers.

A.27. Open problems such as the ones we discussed here create considerable research activity. Many mathematicians attempt to prove conjectures, to find counterexamples, or to understand special cases of famous problems. It is not unusual that a researcher presents an argument to the mathematical community that could possibly prove a conjecture, but then closer inspection shows that his or her results are flawed. For example, in 2004, R. Arenstorf published a draft article on the Internet that, if correct, would have led to a proof of Conjecture A.8. Unfortunately, the French mathematician G. Tenenbaum found an error in this paper that could not be corrected.

A.28. Exercise. Let $k, m, n \in \mathbb{N}$ such that $1 < m \leq k < n$ and $n = mk$. Prove that the Mersenne number M_n is composite. (*Hint:* For $M_4 = 15$ we can write $2^4 - 1 = (2^2 - 1)(2^2 + 1) = 3 \cdot 5$. Think of a similar way to write $2^{mk} - 1$ as a product of two natural numbers distinct from 1.)

A.29. Sometimes $2^n + 1$ is called the n-th Fermat number (rather than $2^{2^n} + 1$), but our definition is more common.

A.30. For many of the problems that we mentioned in this appendix, Hardy and Littlewood gave much more precise statements when they formulated a number of detailed conjectures that are concerned with the distribution of prime numbers. For example they give (conjectured) asymptotic formulae for the number of ways that an even number can be written as a sum of two primes and for the number of twin primes. These statements are often based on heuristic arguments. It is because of arguments of this type that many of the conjectures that we discussed are widely believed to be true.

Further reading

For more details and number-theoretic background, we suggest [**HW**]. Sections 1 and 2 and Appendix 3 of that book are particularly interesting in the context of this appendix, as is [**CP**, Chapter 1]. Readers interested in the Riemann Hypothesis may enjoy the article [**Co**]. The book *Uncle Petros and Goldbach's Conjecture* [**Do**] by Apostolos Doxiadis is a fascinating non-technical book concerning the Goldbach Conjecture.

The article [**LaPe**] is based on new research concerning Sophie Germain's mathematical work and gives deep insight into her ideas on Fermat's Last Theorem and her contribution to number theory generally, which it seems has been widely underestimated. It also sheds some light on the historical context, in particular her correspondence with Gauss.

Finally we would like to mention a webpage that has become quite famous, namely "The Prime Pages" (`http://primes.utm.edu/`). It gives a lot of up-to-date information about various problems concerning prime numbers, open questions, and, for example, the discovery of new very large primes.

Appendix B

Solutions and comments to important exercises

Exercise 1.1.10. (a) Let $n \in \mathbb{N}$ and $M := \{m \in \mathbb{N} : 2m \geq n\}$. Then M is a non-empty subset of $\mathbb{N}$ (as we can see because $n \in M$) and therefore the well-ordering principle yields that M has a smallest element m_0. Hence we have that $2m_0 \geq n$, but $2m < n$ for all natural numbers $m < m_0$. In particular $2m_0 - 2 = 2(m_0 - 1) < n$, and so we conclude that

$$2m_0 - 2 < n \leq 2m_0.$$

This means that n equals $2m_0 - 1$ or $2m_0$. In the first case n is odd in the second case n is even.

If n is even, for example $n = 2m_1$ with a suitable number $m_1 \in \mathbb{N}$, then n cannot be odd at the same time. Assume otherwise and let $m_2 \in \mathbb{N}$ be such that $n = 2m_2 - 1$. Then $2m_1 = 2m_2 - 1$ and in particular $m_1 < m_2$. Then $m_2 - m_1 \in \mathbb{N}$ whereas $m_2 - m_1 = \frac{1}{2} \notin \mathbb{N}$. This is a contradiction. Hence n is *either* even *or* odd.

(b) Suppose that $n, m \in \mathbb{N}$ are even. Then there are $a, b \in \mathbb{N}$ such that $n = 2a$ and $m = 2b$. This implies that $nm = (2a)(2b) = 2(2ab)$. As $2ab \in \mathbb{N}$, it follows that nm is even.

Next suppose that $n, m \in \mathbb{N}$ are odd. Let $a, b \in \mathbb{N}$ be such that $n = 2a - 1$ and $m = 2b - 1$. Then we have that

$$\begin{aligned} nm &= (2a-1)(2b-1) = 4ab - 2b - 2a + 1 \\ &= (4ab - 2b - 2a + 2) - 1 = 2(ab - b - a + 1) - 1. \end{aligned}$$

As $ab - b - a + 1 \in \mathbb{N}$, we deduce that nm is odd.

Exercise 1.1.11. We need to prove that all non-empty subsets of $\mathbb{Z}$ that are bounded from below (from above) have a smallest (a largest) element.

Hence suppose that $M \subseteq \mathbb{Z}$ is non-empty and that M is bounded from below. Let $K \in \mathbb{Z}$ be a lower bound for M; i.e. for all $x \in M$ we have that $K \leq x$. Set

$$S := \{1 + x - K : x \in M\}.$$

For all $x \in M$ we see that $x - K \in \mathbb{N}_0$ and hence $1 + (x - K) \in \mathbb{N}$. It follows that $S \subseteq \mathbb{N}$ and, since M is non-empty, the set S is also non-empty. The well-ordering principle yields that S has a smallest element s_0. Let $m_0 := s_0 - 1 + K$. Then $m_0 \in M$ because of the definition of S. Now we prove that m_0 is the smallest element of M:

Assume that $m \in M$ is such that $m < m_0$. Then $s := 1 + m - K \in S$ and $s = 1 + m - K < 1 + m_0 - K = s_0$, which is a contradiction. Thus m_0 really is the smallest element of M as stated.

Next suppose that $M \subseteq \mathbb{Z}$ is non-empty and bounded from above. Let $K \in \mathbb{Z}$ be an upper bound for M; i.e. for all $x \in M$ we have that $x \leq K$. Then the set $M' := \{-x : x \in M\}$ is a non-empty set of integers that is bounded from below by $-K$. Therefore M' has a smallest element m' by the previous paragraph. Now $m^* := -m'$ is the largest element in M.

These statements do not hold for $\mathbb{Q}$ and $\mathbb{R}$! The set

$$M := \{x \in \mathbb{Q} : 0 < x < 1\}$$

is bounded from above by 1 and from below by 0, but this set has neither a smallest element nor a largest element. (For a formal proof let $x \in M$. Then $0 < \frac{x}{2} < x < \frac{x+1}{2} < 1$ and hence x is neither a smallest element of M nor a largest element.)

Exercise 1.1.12. Let $x \in \mathbb{R}$ be such that $x \neq 1$.

Claim. For all $n \in \mathbb{N}$ the following hold:

(a) $2^n \geq 2n$.

(b) $\displaystyle\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$.

(c) $\displaystyle\sum_{k=0}^{n-1} x^k = \frac{1-x^n}{1-x}$.

(d) $\displaystyle\sum_{k=0}^{n-1} (k+1) \cdot x^k = \frac{nx^{n+1} - (n+1)x^n + 1}{(1-x)^2}$.

Proof of (a). Basis of the induction: As $2^1 = 2 \geq 2 \cdot 1$, our claim is true for $n = 1$.

Induction hypothesis: Suppose that $n \in \mathbb{N}$ is such that the claim holds for n. This means that $2^n \geq 2n$.

Induction step: We have that

$$2^{n+1} = 2 \cdot 2^n \geq 2 \cdot 2n$$

using the induction hypothesis. As $n \geq 1$, we also know that $2n = n + n \geq n + 1$ and hence

$$2^{n+1} \geq 2 \cdot 2n \geq 2 \cdot (n+1).$$

This concludes the proof. ■

Proof of (b). Basis of the induction: The claim holds for $n = 1$ because

$$\sum_{k=1}^{1} k = 1 = \frac{1(1+1)}{2}.$$

Induction hypothesis: Here we go from $n-1$ to n because this simplifies notation. (See (a) in Section 1.1.) Hence suppose that $n \in \mathbb{N}$ is such that $n \geq 2$ and such that the claim holds for $n-1$, i.e.

$$\sum_{k=1}^{n-1} k = \frac{(n-1)n}{2}.$$

Induction step: We split the sum in two parts and then apply the induction hypothesis:

$$\begin{aligned}\sum_{k=1}^{n} k &= \sum_{k=1}^{n-1} k + n = \frac{(n-1)n}{2} + n \\ &= \frac{n^2-n}{2} + \frac{2n}{2} = \frac{n^2-n+2n}{2} = \frac{n^2+n}{2} = \frac{n(n+1)}{2}. \qquad \blacksquare\end{aligned}$$

Proof of (c). Basis of the induction: If $n = 1$, then the left-hand side gives

$$\sum_{k=0}^{n-1} x^k = \sum_{k=0}^{0} x^k = x^0 = 1$$

and the right-hand side gives $\frac{1-x^n}{1-x} = \frac{1-x^1}{1-x} = 1$. These are equal and hence the claim is true for $n = 1$.

Induction hypothesis: Suppose that $n \in \mathbb{N}$ is such that the claim holds for n. This means that

$$\sum_{k=0}^{n-1} x^k = \frac{1-x^n}{1-x}.$$

Induction step: When calculating the left-hand side of the equation for $n+1$, we obtain that

$$\sum_{k=0}^{n} x^k = \sum_{k=0}^{n-1} x^k + x^n = \frac{1-x^n}{1-x} + x^n,$$

applying the induction hypothesis. Thus

$$\begin{aligned}\sum_{k=0}^{n} x^k &= \frac{1-x^n}{1-x} + x^n = \frac{1-x^n+x^n(1-x)}{1-x} \\ &= \frac{1-x^n+x^n-x^{n+1}}{1-x} = \frac{1-x^{n+1}}{1-x}\end{aligned}$$

as stated, finishing the proof. $\blacksquare$

Proof of (d). Basis of the induction: We calculate

$$\sum_{k=0}^{0}(k+1)\cdot x^k = 1\cdot x^0 = 1 \quad \text{and}$$
$$\frac{1\cdot x^2 - 2\cdot x^1 + 1}{(1-x)^2} = \frac{x^2-2x+1}{x^2-2x+1} = 1,$$

so the claim holds if $n=1$.

Induction hypothesis: Let $n \in \mathbb{N}$ be such that

$$\sum_{k=0}^{n-1}(k+1)\cdot x^k = \frac{nx^{n+1}-(n+1)x^n+1}{(1-x)^2}.$$

Induction step: Here we calculate that

$$\sum_{k=0}^{n}(k+1)\cdot x^k = \left(\sum_{k=0}^{n-1}(k+1)\cdot x^k\right) + (n+1)\cdot x^n$$
$$= \frac{nx^{n+1}-(n+1)x^n+1}{(1-x)^2} + (n+1)\cdot x^n = \frac{nx^{n+1}-(n+1)x^n+1+(n+1)x^n(1-x)^2}{(1-x)^2}$$
$$= \frac{nx^{n+1}-(n+1)x^n+1+(n+1)x^n(x^2-2x+1)}{(1-x)^2}$$
$$= \frac{nx^{n+1}-(n+1)x^n+1+(n+1)x^{n+2}-(2n+2)x^{n+1}+(n+1)x^n}{(1-x)^2}$$
$$= \frac{(n+1)x^{n+2}-(n+2)x^{n+1}+1}{(1-x)^2}.$$

Then the proof is complete.

(The readers who know about differential calculus may alternatively take the statement in (c) and then differentiate on both sides.) ■

Remark. The remaining parts of this exercise are proved similarly, but we do not explain the details because the statements are not relevant for the content of this book.

Exercise 1.1.15. Suppose that $k, n \in \mathbb{N}_0$ are such that $k \leq n$. Following our intuitive definition, the binomial coefficient $\binom{n}{k}$ counts how many subsets of size k we find in the set $\{1,\ldots,n\}$.

Here is a way to make this explicit: we write the numbers $1,\ldots,n$ in some arbitrary order and then take the first k numbers. There are $n!$ ways to write $1,\ldots,n$ in arbitrary order. If we write $1,\ldots,n$

in two ways such that the first k numbers are the same (even if in different order), then the other $n-k$ numbers must coincide as well (again independent of their ordering). This means that, whenever we choose k numbers in the set $\{1,\ldots,n\}$, then this choice corresponds to exactly $k!(n-k)!$ sequences built from the numbers $1,\ldots,n$. Hence the number of possibilities of choosing k numbers from $\{1,\ldots,n\}$ is exactly $n!$ divided by $k!(n-k)!$, or in other words,

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}. \tag{B.1}$$

Now we prove (B.1) by using the recursive definition of binomial coefficients. Recall that

$$\binom{0}{0} = 1, \quad \binom{0}{k} = 0 \ (k \neq 0), \quad \text{and} \quad \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$$

for all $n, k \in \mathbb{N}_0$.

Basis of the induction: If $n = 0$, then $\binom{0}{0} = 1$ and $\frac{0!}{0!\cdot 0!} = 1$, so the claim holds.

Induction hypothesis: Suppose that $n \in \mathbb{N}$ is such that $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ for all $k \leq n$.

Induction step: Let $k \in \mathbb{N}_0$ be such that $k \leq n+1$. If $k = 0$ or $k = n+1$, then

$$\binom{n+1}{k} = 1 = \frac{(n+1)!}{0!(n+1)!} = \frac{(n+1)!}{k!(n+1-k)!}.$$

Hence we suppose from now on that $1 \leq k \leq n$. The induction hypothesis and the recursive formula yield that

$$\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} = \frac{n!}{k!(n-k)!} + \frac{n!}{(k-1)!(n-(k-1))!}.$$

Hence it follows that

$$\begin{aligned}\binom{n+1}{k} &= \frac{n!}{k!(n-k)!} + \frac{n!}{(k-1)!(n-(k-1))!}\\ &= \frac{(n-k+1)\cdot n! + k\cdot n!}{k!(n-k+1)!}\\ &= \frac{n\cdot n! - k\cdot n! + n! + k\cdot n!}{k!(n-k+1)!}\\ &= \frac{n\cdot n! + n!}{k!(n-k+1)!} = \frac{(n+1)!}{k!(n-k+1)!},\end{aligned}$$

and the proof is complete.

Exercise 1.1.17. Let $n, k, \ell \in \mathbb{N}_0$.

Claim. Then the following hold:

(a) $\binom{n+\ell}{k} \geq \binom{n}{k}$;

(b) $\binom{n+\ell}{k+\ell} \geq \binom{n}{k}$;

(c) $\binom{2n}{n} \geq 2^n$.

Sketch of proof. A formal proof of (a) and (b) could for example use induction over ℓ and the recursive formula that we have for binomial coefficients. But it can be seen that the statements are true by looking at Pascal's triangle; each entry is the sum of the two entries above, and all entries are non-negative.

For (c) we use induction and parts (a) and (b). If $n = 0$, then the inequality is correct because $\binom{0}{0} = 1$ and $2^0 = 1$. This establishes the induction basis. Now we suppose that the claim holds for some $n \in \mathbb{N}$ and we calculate

$$\begin{aligned}\binom{2(n+1)}{n+1} = \binom{2n+1}{n} + \binom{2n+1}{n+1} &\geq \binom{2n}{n} + \binom{2n}{n}\\ = 2\cdot\binom{2n}{n} &\geq 2\cdot 2^n = 2^{n+1}.\end{aligned}$$

In the first line we use the recursive definition and (a) and (b). In the second line we use the induction hypothesis. ■

Exercise 1.1.18. Let $n, m \in \mathbb{N}_0$. Let $a(n,m)$ denote the number of possibilities for choosing m integers between 1 and n, not necessarily pairwise distinct, but ordered. We prove that $a(n,m)$ satisfies the following recursive formula:

$$a(n,m) = a(n-1,m) + a(n,m-1).$$

To see this, we choose up to m distinct numbers between 1 and n. If n is one of the chosen numbers, then there is one place less for the other numbers, so the number of possibilities that this happens is $a(n,m-1)$. (Recall that the chosen numbers are not necessarily distinct, so n could be chosen several times.) If n is not among the chosen numbers, then we might as well have chosen between 1 and $n-1$, so the number of possibilities for this is $a(n-1,m)$. As one or the other case must occur, the number of all possibilities is the sum of these numbers.

Claim. For all $n, m \in \mathbb{N}_0$ we have that $a(n,m) = \binom{n+m}{m}$.

Proof. We argue by induction over $n+m$. Hence for all $\ell \in \mathbb{N}_0$ we prove that if $n, m \in \mathbb{N}_0$ satisfy $n+m=\ell$, then

$$a(n,m) = \binom{n+m}{m}.$$

Basis of the induction: We see that $a(0,0) = 1 = \binom{0}{0}$, so the claim holds if $\ell = 0$.

Induction hypothesis: Let $\ell \geq 1$ be such that the claim holds for $\ell - 1$. Hence if $m, n \geq 1$ are such that $m+n=\ell$, then

$$a(n-1,m) = \binom{n+m-1}{m} \quad \text{and} \quad a(n,m-1) = \binom{n+m-1}{m-1}.$$

Induction step: Suppose that $m, n \in \mathbb{N}_0$ are such that $\ell = m+n$. If $m=0$ or $n=0$, then $a(n,m)=1$ and $\binom{n+m}{m} = 1$ and hence there is nothing left to prove.

Otherwise we apply the recursive formula for $a(n,m)$, the induction hypothesis, and the recursive formula for binomial coefficients.

Then

$$\begin{aligned} a(n,m) &= a(n-1,m) + a(n,m-1) \\ &= \binom{n+m-1}{m} + \binom{n+m-1}{m-1} = \binom{n+m}{m}. \end{aligned} \qquad \blacksquare$$

Exercise 1.2.6.

Claim. Let $n \in \mathbb{N}$.

(a) If $n > 1$, then there exists a prime number p dividing n.

(b) If $n > 1$ is composite, then there exists a non-trivial divisor k of n such that $k^2 \leq n$.

Proof of (a). As $n > 1$, there are two possibilities:

- n is prime or
- n is composite.

In the first case we set $p := n$. In the second case we consider $T := \{k \in \mathbb{N} : k > 1 \text{ and } k \,|\, n\}$. This set is non-empty because it contains n. The well-ordering principle yields a smallest element k_0 of T. Suppose that $m \neq 1$ is a number that divides k_0. Then m divides n as well and therefore $m \in T$. The choice of k_0 as smallest element of T forces $m = k_0$. Thus we proved that k_0 has exactly two distinct divisors, namely 1 and k_0. This means that k_0 is prime and we set $p := k_0$. $\blacksquare$

Proof of (b). Suppose that $n > 1$ is composite. Then let $a, b \in \mathbb{N}$ be such that $a, b \neq 1$ and $n = a \cdot b$. If $a \leq b$, then $a^2 \leq a \cdot b = n$ and hence a is as claimed. Otherwise $b < a$ and then b is as claimed by the same argument. $\blacksquare$

Exercise 1.2.12. Let $n \in \mathbb{N}$ and let p be a prime number that does not divide n. We set $d := \gcd(p, n)$. Then d divides p and therefore $d = p$ or $d = 1$ because p is prime. In the first case it follows that p divides n, contrary to our hypothesis. Hence $d = 1$ as claimed.

Exercise 1.3.9. Let $a, b \in \mathbb{Z}$. Set $d := \gcd(a, b)$ and $k := \operatorname{lcm}(a, b)$.

Claim. (a) $\frac{a}{d}$ and $\frac{b}{d}$ are coprime.

(b) If v is a common multiple of a and b, then $k \mid v$.

(c) $d \cdot k = |a \cdot b|$. In particular, if $d = 1$, then $k = |a \cdot b|$.

Proof of (a). Let $m \in \mathbb{N}$ be a common factor of $\frac{a}{d}$ and $\frac{b}{d}$. Then $m \cdot d$ is a common factor of a and b. As d is the greatest common divisor of a and b, it follows that $m \cdot d \leq d$ and hence $m = 1$. Now it follows that $\gcd\left(\frac{a}{d}, \frac{b}{d}\right) = 1$. ■

Proof of (b). We divide v by k with remainder, so we write

$$v = q \cdot k + r$$

such that $q \in \mathbb{N}_0$ and $0 \leq r < k$. We write this differently as

$$r = v - q \cdot k.$$

Since a is a factor of v and k, it also divides r. Moreover b divides r. As r is a common multiple of a and b, the definition of $k = \operatorname{lcm}(a, b)$ and the choice of r yield that $r \notin \mathbb{N}$. Thus $r = 0$. It follows that $v = q \cdot k$ and hence $k \mid v$ as stated. ■

Proof of (c). Here we need to show that

$$d = \frac{|a \cdot b|}{k}.$$

First we observe that $|a \cdot b|$ is a common multiple of a and b, and so (b) yields that k divides $|a \cdot b|$. This means that $m := \frac{|a \cdot b|}{k}$ is a natural number. Now

$$\frac{a \cdot b}{d} = a \cdot \frac{b}{d} = b \cdot \frac{a}{d}$$

is a common multiple of a and b. Therefore $\frac{|a \cdot b|}{d} \geq k$ and we deduce that $d \leq m$.

We want to prove that m is a common factor of a and b. So we write $k = a \cdot b_1$ with some $b_1 \in \mathbb{Z}$. Then

$$m = \frac{a \cdot b}{k} = \frac{a \cdot b}{a \cdot b_1} = \frac{b}{b_1},$$

and this implies that $b = m \cdot b_1$. Thus m divides b.

In a similar way we see that m divides a. Consequently m is a common factor of a and b, $m \geq d$, and $m \in \mathbb{N}$. Our definition of the greatest common divisor gives that $m = d$, as claimed. ■

Exercise 2.3.4.

Claim (a). Suppose that $f : \mathbb{N} \to \mathbb{R}$ is a function and that $\varepsilon > 0$ satisfies $f(n) > \varepsilon$ for all $n \in \mathbb{N}$. Let $C \in \mathbb{R}$ be a constant.

Then $f(n) + C = O(f(n))$.

Proof. For all $n \in \mathbb{N}$ we have that

$$|f(n)+C| \leq |f(n)|+|C| = |f(n)| \cdot \left(1 + \frac{C}{|f(n)|}\right) < |f(n)| \cdot \left(1 + \frac{C}{\varepsilon}\right).$$

So if we set $K := 1 + \frac{C}{\varepsilon}$, then $|f(n) + C| < K \cdot |f(n)|$ for all $n \in \mathbb{N}$. Then the concept of the O-notation yields that we are done. ■

Claim (b). Let $k, m \in \mathbb{N}_0$. Then $x^k = O(x^m)$ if and only if $k \leq m$.

Proof. First suppose that $k \leq m$. Then $x^k \leq x^m$ for all $x \in \mathbb{N}$ and hence $x^k = O(x^m)$.

Conversely suppose that $k > m$ and let C be a constant. If $x > C$, then

$$x^k \geq x^{m+1} = x \cdot x^m > C \cdot x^m.$$

Hence $x^k \neq O(x^m)$. ■

Claim (c). If P is an integer polynomial of degree at most d, then $P(n) = O(n^d)$.

Proof. We let $a_0, \ldots, a_d \in \mathbb{Z}$ be such that $P = a_d X^d + \cdots + a_1 X + a_0$. Then for all $n \in \mathbb{N}$ we see that

$$P(n) = a_d n^d + \cdots + a_1 n + a_0 \leq (a_d + \cdots + a_1 + a_0) \cdot n^d.$$ ■

Claim (d). $a^n = O(2^n)$ if and only if $a \leq 2$.

Proof. If $a \leq 2$, then $a^n \leq 2^n$ and hence $a^n = O(2^n)$. Now we suppose that $a > 2$ and we set $b := \frac{a}{2}$. Then

$$a^n = (b \cdot 2)^n = b^n \cdot 2^n.$$

Assume that $a^n = O(2^n)$. Then $b^n \leq C$ for some constant $C > 0$. But $b > 1$ and hence $n \leq \frac{\log b}{\log C}$, and this is false if n is large. This contradiction shows that $a^n \neq O(2^n)$ in this case. ■

Claim (e). Suppose that $\varepsilon > 0$ is a real number. Then $\log n = O(n^\varepsilon)$.

Proof. First we write

$$\log n = \log\big((n^\varepsilon)^{\frac{1}{\varepsilon}}\big) = \frac{\log(n^\varepsilon)}{\varepsilon}.$$

Then we see that $\log x < x$ for all numbers $x \in \mathbb{R}$ such that $x > 0$. This is a consequence, for example, of Exercise 1.1.12(a) where we proved that $2^n \geq 2n$ for all $n \in \mathbb{N}$. We conclude that $n \geq \log n + 1$ and thus

$$\log x \leq \log\lceil x\rceil \leq \lceil x\rceil - 1 < x.$$

Now

$$\log n = \frac{\log(n^\varepsilon)}{\varepsilon} < \frac{n^\varepsilon}{\varepsilon}$$

and therefore $\log n = O(n^\varepsilon)$, as stated. ■

Claim (f). If $k \in \mathbb{N}$, then $n^k = O(2^n)$.

Proof. Let $k \in \mathbb{N}$. First we show that, for all sufficiently large $n \in \mathbb{N}$, the inequality $(n + 1)^k < 2n^k$ holds. Recall that, by the binomial theorem, we may write $(n + 1)^k$ as $n^k + P(n)$ where

$$P(n) := \sum_{j=0}^{k-1} \binom{k}{j} n^j$$

is an integer polynomial of degree $k - 1$ in n. With (c) we see that $P(n) \leq K \cdot n^{k-1}$ for some suitable $K > 0$. Now if $n > K$, then

$$(n + 1)^k = n^k + P(n) \leq n^k + K \cdot n^{k-1} < n^k + n^k = 2n^k.$$

Next we let $n_0 := \lceil K\rceil$ and $C := \frac{n_0^k}{2^{n_0}}$. We claim that

$$n^k \leq C \cdot 2^n$$

for all $n \geq n_0$. For n_0 itself this follows from the definition of C. Proceeding by induction we suppose that the claim holds for some n. Then

$$(n + 1)^k < 2n^k \leq 2 \cdot C \cdot 2^n = C \cdot 2^{n+1}.$$ ■

Exercise 2.3.5. We sketch an argument that shows the efficiency of long division with remainder. Let $n, k \in \mathbb{N}$ and, for simplicity, suppose that they are represented in the binary system. If $n \leq k$, then we choose n as remainder and we are done. Hence suppose that $n > k$. By s we denote the number of digits of n and by t the number of digits of k. Our aim is to find $q, r \in \mathbb{N}_0$ such that $n = q \cdot k + r$ and $0 \leq r < k$. The numbers q and r are found after $s - t + 1$ steps in the following way: we set $r_0 = n$ and $r_{s-t+1} = r$, and for all $j \in \{0, \ldots, s - t + 1\}$ we calculate the j-th digit of q and some remainder r_j.

- We calculate $k'_j := 2^{s-t+1-j} \cdot k$ by adding $s - t + 1 - j$ zeros to the number k.
- If $k'_j \geq r_{j-1}$, then at the j-th digit q has the entry 1. Then we set $r_j := r_{j-1} - k'_j$.
- Otherwise the j-th digit of q is 0 and then we set $r_j := r_{j-1}$.

We have to compare at most $s - t + 1$ digits and then we subtract numbers with at most s digits. Hence the running time is polynomial in s.

Remark. By using the method of "divide and conquer", it is possible to give an easier but less efficient algorithm. There we search for the largest number m such that $k \cdot m \leq n$. As $m \leq n$ this search needs at most $\lceil \log m \rceil$ multiplications.

Now we consider the Euclidean algorithm. Applied to two natural numbers m and n such that $m > n$, we begin by dividing m by n with remainder:

$$m = q_1 \cdot n + r_2, \quad q_1 \geq 1, \quad r_2 < n.$$

Then $r_2 < \frac{m}{2}$; this follows immediately if $n \leq \frac{m}{2}$ because then $r_2 < n$. Otherwise $q_1 = 1$ and again

$$r_2 = m - n < m - \frac{m}{2} = \frac{m}{2}.$$

Hence for all numbers r_j that appear during the Euclidean algorithm, we have that $r_{j+2} < \frac{r_j}{2}$ and in particular

$$r_{2k} < \frac{m}{2^k}.$$

Thus the number of divisions that need to be performed until the remainder is 0 is at most $2\cdot\lceil\log m\rceil - 1$. Hence the order of magnitude is $O(\log m)$ and the algorithm is efficient.

Exercise 2.3.6. Let $k, n \in \mathbb{N}$. We want to calculate the number $m_0 := \lfloor\sqrt[k]{n}\rfloor$ efficiently and hence we apply the idea of "divide and conquer" again. We see that $m_0 \leq n$ and hence we need to check at most $\lceil\log n\rceil$ times whether or not a power m^k is larger than n. The previous exercise yields that this can be done efficiently. (We stop taking powers as soon as we reach a number that is larger than n and hence our calculations only deal with numbers that have at most $2\lfloor\log n\rfloor + 2$ digits.) Hence the algorithm is efficient.

When it comes to practical applications, the algorithm could be improved by finding better bounds for m_0. Recall that the number t of digits of n in the binary system is exactly $\lfloor\log n\rfloor + 1$. Then the definition of m_0 implies that $m_0^k \leq n$ and $(m_0+1)^k > n$. It follows that

$$\log m_0 \leq \left\lceil\frac{t}{k}\right\rceil \quad \text{and} \quad \log(m_0+1) > \left\lfloor\frac{t-1}{k}\right\rfloor;$$

therefore

$$2^{\left\lfloor\frac{t-1}{k}\right\rfloor} - 1 \leq m_0 \leq 2^{\left\lceil\frac{t}{k}\right\rceil}.$$

Consequently the idea of "divide and conquer" needs to be applied only to m_0.

When checking whether or not, for a natural number n, there exist $m, k \in \mathbb{N}$ such that $k > 1$ and $n = m^k$, for all $k \in \{2, 3, 4, \dots\}$ we could just calculate the numbers $m_k := \lfloor\sqrt[k]{n}\rfloor$. If it happens that $m_k^k = n$, then we answer "yes", and if $m_k^k > n$, then we answer "no".

We need to check at most $\log n$ numbers k; thus the algorithm is efficient.

Exercise 2.5.5.

Claim. Suppose that P is an integer polynomial in n variables that is not the zero polynomial. Let d be the highest exponent that appears for one of the variables and let $M > 0$. Then P has at most $n\cdot d\cdot M^{n-1}$ integer zeros (i.e. zeros with integer coordinates) with coordinates between 1 and M.

Proof. We argue by induction over n. If $n = 0$, then P has no variables at all, so it is constant and non-zero. This implies that P has no zeros at all.

Next we suppose that $n \in \mathbb{N}$ is such that the claim holds for n. Let $y_1, \dots, y_n, y_{n+1} \in \mathbb{Z}$ be such that $P(y_1, \dots, y_n, y_{n+1}) \neq 0$. (This choice is possible because P is not the zero polynomial.)

Suppose that $(x_1, x_2, \dots, x_n, x_{n+1})$ is a zero of P with integer coordinates $x_j \in \{1, \dots, M\}$ for all $j \in \{1, \dots, n+1\}$.

There are two cases:

(a) If $P(y_1, y_2, \dots, y_n, x_{n+1}) \neq 0$, then $(x_1, \dots, x_n)$ is a zero of a non-constant polynomial in n variables $X_1, \dots, X_n$, namely

$$Q(X_1, \dots, X_n) := P(X_1, \dots, X_n, x_{n+1}).$$

By our induction hypothesis Q has at most $n \cdot d \cdot M^{n-1}$ integer zeros with coordinates between 1 and M. As x_{n+1} takes values between 1 and M as well, we find at most $n \cdot d \cdot M^n$ zeros of this kind.

(b) If $P(y_1, \dots, y_n, x_{n+1}) = 0$, then x_{n+1} is a zero of

$$R(X) := P(y_1, \dots, y_n, X).$$

Thus Corollary 3.4.5 implies that there are at most d such values x_{n+1}. There are M^n possibilities for $x_1, \dots, x_n$; hence we find $d \cdot M^n$ zeros of this kind.

This case distinction shows that there are at most

$$n \cdot d \cdot M^n + d \cdot M^n = (n+1) \cdot d \cdot M^n$$

zeros of the required form. ■

Exercise 2.5.6. We consider a coin that gives “heads” with probability p and “tails” with probability $q = 1 - p$.

Claim (a). The probability of no “tails” after throwing n times is q^n.

Proof. It is one of the standard arguments in probability theory that the probability for the occurrence of n independent events is the product of the individual probabilities. ■

Claim (b). If $q = 1/2$, then we need at least 20 throws to ensure that the probability from (a) is at most 0.0001%.

Proof. By (a), the probability of no “tails” after throwing n times is $1/2^n$. If we want to force

$$\frac{1}{2^n} < 0.000001,$$

then we need to choose n such that

$$n = \left\lceil \log \frac{1}{0.000001} \right\rceil = \lceil \log 10^6 \rceil = 20.$$

This means that after throwing the coin 20 times, we will obtain the result “heads” at least once with probability 0.999999. ■

Claim (c). The average number of throws until we have the event “heads” for the first time is

$$\frac{1}{p}.$$

Proof. The probability that “heads” appears for the first time after throwing n times is $p \cdot q^{n-1}$ because it means that we throw “tails” $n-1$ times first.

Hence the expected value is

$$\sum_{j=1}^{\infty} j \cdot p \cdot q^{j-1} = p \cdot \sum_{j=0}^{\infty} (j+1) \cdot q^j.$$

Using Exercise 1.1.12, we calculate this sum as

$$\sum_{j=0}^{k-1} (j+1) \cdot q^j = \frac{kq^{k+1} - (k+1)q^k + 1}{(1-q)^2}.$$

When k goes to ∞, then the fact that $q < 1$ yields that the numerator on the right-hand side tends to 1. Thus

$$p \cdot \sum_{j=0}^{\infty} (j+1) \cdot q^j = \frac{p}{(1-q)^2} = \frac{p}{p^2} = p,$$

as stated. ■

In the algorithm POLY-NULL, the probability of finding a nonzero is at least 1/2. Hence, on average, we need at most two repetitions of the algorithm to see that a polynomial is not the zero polynomial.

Exercise 3.1.10.

Claim. Suppose that $a, b \in \mathbb{Z}$ and that $m, n \in \mathbb{N}$ are such that $m \mid n$ and $a \equiv b \pmod{n}$. Then $a \equiv b \pmod{m}$. The converse is false.

Proof. The hypothesis $a \equiv b \pmod{n}$ means that n divides $a - b$. As m divides n, this implies that $a - b$ is divisible by m and thus $a \equiv b \pmod{m}$. We give a counterexample for the converse: 2 divides 4 and $8 \equiv 10 \pmod{2}$, but 8 is not congruent to 10 modulo 4. ■

Exercise 3.1.13. The numbers 4 and 3 are zero divisors modulo 6 because they are both incongruent to 0 modulo 6, but their product $3 \cdot 4 = 12$ is divisible by 6.

Also 2 and 5 are zero divisors modulo 10 because they are incongruent to 0 modulo 10, but $2 \cdot 5 = 10$ is congruent to 0 modulo 10.

Now let p be a prime number and let $x, y \in \mathbb{Z}$ be such that $xy \equiv 0 \pmod{p}$. Then p divides xy and hence it divides one of x or y, by Corollary 1.3.5. This means that there are no zero divisors modulo a prime number.

Claim. Let $n \in \mathbb{N}$ be such that $n \geq 2$. Then there are zero divisors modulo n if and only if n is composite.

Proof. We proved in the paragraph above that there are no zero divisors modulo a prime. Hence if there are zero divisors modulo n, then n is composite. For the converse we suppose that n is composite. Then choose $x, y \in \mathbb{N}$ such that $n = xy$ and $1 < x \leq y < n$. Neither x nor y is divisible by n, but $xy = n$ is, and therefore x and y are zero divisors modulo n. ■

Exercise 3.1.14. Let $a, n, k \in \mathbb{N}$ be such that $n \geq 2$. We may suppose that $a < n$; otherwise we divide a by n with remainder. This can be done efficiently. In order to calculate the remainder of a^k modulo n, we argue as in Section 2.3, when we discussed the method “divide and conquer” for computing powers. But here, all numbers that occur will be viewed modulo n. This means that at most $2\lfloor \log k \rfloor$ multiplications are necessary, and all these numbers are at most of size n. Hence the algorithm is efficient.

Now we calculate the multiplicative inverse of a modulo n. In Exercise 2.3.5 we saw that the Euclidean algorithm is efficient and that, when applied to n and a, it needs at most $2\lceil \log n \rceil - 1$ divisions with remainder. Substituting backwards to find the numbers from Bézout’s Lemma takes at most $\log n$ steps and all these numbers can be viewed modulo n. Hence all numbers that appear in the calculation have at most as many digits as n and the number of calculations is bounded by a polynomial in $\log n$. Thus the algorithm is efficient.

Remark. The numbers s and t such that $s \cdot n + t \cdot a = 1$ that are found by the Euclidean algorithm have size at most n. Thus we do not really need to reduce the numbers modulo n in each step; but we will not prove this fact here.

Exercise 3.2.13. Let $a, n \in \mathbb{Z}$, $n \geq 2$, and suppose that a and n are coprime. Set $k := \operatorname{ord}_n(a)$.

Claim. (a) If $b_1, b_2 \in \mathbb{N}_0$ are such that $b_1 \equiv b_2 \pmod{k}$, then $a^{b_1} \equiv a^{b_2} \pmod{n}$.

(b) Let $A := \{a^j \bmod n : j \geq 0\}$ be the set of remainders modulo n of all powers of a. Then

$$A = \{1, a \bmod n, a^2 \bmod n, \ldots, a^{k-1} \bmod n\}.$$

(c) $\#A = k$.

Proof. For (a) we let $b_1, b_2 \in \mathbb{N}_0$ be such that $b_1 \equiv b_2 \pmod{k}$. We may suppose that $b_1 \leq b_2$.

By hypothesis k divides $b_2 - b_1$ and thus there is some $s \in \mathbb{N}_0$ such that $b_2 - b_1 = k \cdot s$. Now

$$a^{b_1} = a^{ks+b_2} = (a^k)^s \cdot a^{b_2} \equiv 1^s \cdot a^{b_2} = a^{b_2} \pmod{n}$$

as stated.

This already implies that the set A defined in (b) has at most k distinct elements and that these elements are precisely those from (b). What remains to prove (c) is that the powers $1, a, a^2, ..., a^{k-1}$ are distinct modulo n. This follows from Lemma 3.2.1. ■

Exercise 3.2.17.

Claim. (a) If $n, m \in \mathbb{N}$ are coprime, then $\varphi(nm) = \varphi(n) \cdot \varphi(m)$.

(b) If p is prime and if $k \in \mathbb{N}$, then $\varphi(p^k) = (p-1) \cdot p^{k-1}$.

Proof of (a). The Chinese Remainder Theorem (Theorem 3.1.7) says that if $a_1, a_2 \in \mathbb{N}_0$ are such that $a_1 < n$ and $a_2 < m$, then there is a unique natural number x between 0 and $nm - 1$ such that $x \equiv a_1 \pmod{n}$ and $x \equiv a_2 \pmod{m}$.

If x and nm are coprime, then x is also coprime to n and to m, i.e. $a_1 \in \text{cp}(n)$ and $a_2 \in \text{cp}(m)$. Conversely if a_1 is coprime to n and a_2 is coprime to m, then x is coprime to n and to m. Hence x is coprime to nm by Corollary 1.3.5.

Hence there are exactly $\varphi(n) \cdot \varphi(m)$ possibilities for choosing the numbers a_1 and a_2 in such a way that x is coprime to nm. ■

Proof of (b). Those numbers in $\{1, ..., p^k\}$ that are *not* coprime to p^k are precisely the multiples of p that are contained in this set, i.e. the numbers

$$p, 2p, ..., p^2, p^2 + p, ..., 2p^2, ..., p^{k-1}, p^{k-1} + p, ...$$

The number of these powers of p is p^{k-1}, and all remaining numbers from 1 to $p^k - 1$ are coprime to p^k. Therefore $\varphi(p^k) = p^k - p^{k-1} = (p-1) \cdot p^{k-1}$. ■

Now we apply the results that we have just proved and we calculate $\varphi(10)$, $\varphi(50)$, and $\varphi(180)$:

$$\varphi(10) = \varphi(2 \cdot 5) = \varphi(2)\varphi(5) = 1 \cdot 4 = 4;$$
$$\varphi(50) = \varphi(2 \cdot 5^2) = \varphi(2)\varphi(5^2) = 1 \cdot 4 \cdot 5^1 = 20;$$
$$\varphi(180) = \varphi(2^2)\varphi(3^2)\varphi(5) = (1 \cdot 2) \cdot (2 \cdot 3) \cdot 4 = 2 \cdot 6 \cdot 4 = 48.$$

Claim. If $n \in \mathbb{N}$ and $n > 2$, then $\varphi(n)$ is even.

Proof. First we suppose that n possesses an odd prime factor p. Then let $k, m \in \mathbb{N}$ be such that $p \nmid m$ and $n = p^k \cdot m$. Now we see that

$$\varphi(n) = \varphi(p^k \cdot m) = \varphi(p^k) \cdot \varphi(m) = (p-1) \cdot p^{k-1} \cdot \varphi(m).$$

Then $p - 1$ is even because p is odd, and thus $\varphi(n)$ is even.

Next suppose that n does not have any odd prime factors. Then n is a power of 2, which means that there is some $k \in \mathbb{N}$ such that $k \geq 2$ and $n = 2^k$. (We recall that $n > 2$.) It follows that

$$\varphi(n) = \varphi(2^k) = (2-1) \cdot 2^{k-1} = 2^{k-1}.$$

As $k \geq 2$, this is again an even number. ■

Exercise 3.4.16.

Claim. Suppose that P is a rational polynomial of degree at least 1. Then there exists an integer polynomial H that is irreducible over $\mathbb{Q}$ and divides P.

Proof. Our arguments resemble those that we used when proving that all natural numbers of size at least 2 possess a prime factor.

Let $k \geq 1$ be minimal with the property that there exists a rational polynomial H' of degree k that divides P over $\mathbb{Q}$. The coefficients of H' are rational and we let d denote the lowest common multiple of their denominators. Then $H := d \cdot H'$ is an integer polynomial of degree k. As $H' = \frac{1}{d} \cdot H$, it follows that H divides P over $\mathbb{Q}$.

We claim that H is irreducible over $\mathbb{Q}$. Assume that T is a nontrivial divisor of H over $\mathbb{Q}$. Then T also divides P and $1 \leq \deg T < k$, so this contradicts our choice of k. ■

Exercise 3.5.9. Suppose that $n \geq 2$ is a natural number. Moreover let P and Q be polynomials such that the leading coefficient of P is coprime to n and, for simplicity, all coefficients of P and Q are between 0 and $n-1$. We consider long division with remainder modulo n, so we are looking for polynomials T and Q such that

$$Q \equiv T \cdot P + R \pmod{n}.$$

Here is how we proceed:

1. If $\deg Q < \deg P$, then set $T = 0$ and $R = Q$.
2. Otherwise we let $k := \operatorname{grad} Q - \operatorname{grad} P$ and we divide the leading coefficient of Q modulo n by the leading coefficient of P. The result is denoted by a. Then the polynomial T will have degree k and leading coefficient a.
3. For the remaining coefficients of T and for the remaining polynomial R we calculate $Q' := Q - a \cdot P \cdot X^k$ and we reduce all coefficients modulo n. Then we replace Q by Q' and we go back to Step 1.

We know that division modulo n can be performed efficiently. Step 3 needs at most one multiplication and one subtraction modulo n for each coefficient of Q. As Q' has smaller degree than Q modulo n, Steps 2 and 3 are done at most $(\deg Q - \deg P)$ times.

We conclude that this procedure has runtime polynomial in $\log n$, $\deg P$, and $\deg Q$. In particular the calculation of sums and products modulo n and H can be performed efficiently because sums and products of integer polynomials can be done efficiently. Finally, powers modulo n and H can be calculated as in Exercise 3.1.14. In each step we need to replace the polynomials that appear by their remainder when dividing by H and reducing modulo n. This way we avoid large coefficients and large degrees.

Exercise 3.5.11. Let $n \geq 2$, $a \in \mathbb{Z}$ and let P be a polynomial.

Claim. (a) a is a zero of P modulo n if and only if $(X-a)$ divides P modulo n.

(b) If $P \not\equiv 0 \pmod{n}$, then P can be written as

$$P \equiv (X - a_1) \cdots (X - a_m) \cdot Q \pmod{n}. \tag{B.2}$$

Here $m \geq 0$, all numbers $a_1, \ldots, a_m$ are between 0 and $n-1$, and Q is a polynomial that has no zeros modulo n.

(c) If n is prime, then the numbers $a_1, \ldots, a_m$ in (b) are unique up to ordering. This means that if

$$P \equiv (X - b_1) \cdots (X - b_k) \cdot R \pmod{n}$$

is a similar presentation of P modulo n, then $m = k$ and the numbers b_j coincide with the numbers a_j up to ordering.

(d) If n is prime and $P \not\equiv 0 \pmod{n}$, then the number of zeros of P modulo n is at most $\deg_n(P)$.

Sketch of proof. Part (a) is similar to Theorem 3.4.4. If $(X - a)$ divides P modulo n, then a is a zero of P modulo n. Conversely if a is a zero of P modulo n, then we divide P by $(X - a)$ with remainder. The remainder is a constant polynomial and hence it must be congruent to the zero polynomial modulo n.

Part (b) follows from (a) by induction over the degree of P. If P has no zeros modulo n, then $m := 0$ and $Q := P$. Otherwise let a be a zero of P modulo n. Then by (a) we may divide by $X - a$ and we obtain a polynomial P' such that $P \equiv (X - a) \cdot P' \pmod{n}$ and $\deg P' = \deg P - 1$. Then the result follows inductively.

The statement in (c) can be proved with arguments similar to those for Theorem 1.3.2. We write P as in (B.2) and then we argue by induction over m. If $m = 0$, then $P \equiv Q \pmod{n}$ and hence P has no zeros modulo n. Therefore P cannot be written in another way. Otherwise there exists a zero a of P modulo n such that $0 \le a < n$. If

$$P \equiv (X - a_1) \cdots (X - a_m) \cdot Q \pmod{n} \quad \text{and}$$
$$P \equiv (X - b_1) \cdots (X - b_k) \cdot R \pmod{n}$$

are two ways of writing P modulo n as above, then the fact that there are no zero divisors modulo n (see Exercise 3.1.13) yields that one of the numbers a_j and one of the numbers b_j are congruent to a modulo n. By rearranging we may suppose that $a_1 \equiv a \equiv b_1 \pmod{n}$. Now the induction hypothesis yields that

$$P' := (X - a_2) \cdots (X - a_m) \cdot Q \equiv (X - b_2) \cdots (X - b_k) \cdot Q \pmod{n},$$

so the proof is complete.

Now (d) follows from (b) and the fact that there are no zero divisors modulo n (because n is prime). Here the zeros of P modulo n are exactly the numbers congruent to $a_1, \ldots, a_m$ modulo n, and moreover $m \le \operatorname{grad} P$. (Alternatively we could argue as in Corollary 3.4.5 by applying (a) and induction.) ■

Exercise 3.5.15. Suppose that p is a prime number and that P is a polynomial such that $P \not\equiv 0 \pmod{p}$.

Claim.

(a) If $\deg_p(P) > 0$, then there exists a polynomial H that is monic and irreducible modulo p and that divides P modulo p.

(b) There are $m \geq 0$, $a \in \mathbb{Z}$ and some irreducible monic polynomials $H_1, \ldots, H_m$ such that

$$P \equiv a \cdot H_1 \cdots H_m \pmod{p}.$$

Sketch of proof. The proof of (a) is similar to Exercise 3.4.16. Then (b) follows from (a) by induction over the degree of P. We leave the details to the reader. ■

Exercise 4.5.8. Deciding whether n is even or odd only takes a division by 2, so this can be done efficiently. It is even easier if n is given in binary presentation because then we only need to check whether the last digit is a 0. We also saw in Exercise 2.3.6 that powers can be recognized efficiently. Writing the number $n-1$ as $d \cdot 2^l$ takes at most $\lfloor \log n \rfloor + 1$ divisions with remainder. Again this is even easier in the binary system, but we leave these arguments to the reader.

When checking whether or not a and n are coprime, we apply the Euclidean algorithm. It is efficient by Exercise 2.3.5. The calculation of the power $a^d \bmod n$ can also be done efficiently, as we saw in Exercise 3.1.14. The same holds for the powers b, b^2, b^4, etc., where we need to calculate at most $l-1$ numbers. As $l \leq \log n$, this is also efficient and hence the algorithm in total is efficient.

Exercise 5.1.8. Suppose that $n \geq 2$ is a natural number and that p is a prime factor of n. Let j denote the highest exponent such that p^j divides n.

Claim (a). $\binom{n}{p} \not\equiv 0 \pmod{p^j}$.

Proof. We have that

$$p!\binom{n}{p} = \frac{n!}{(n-p)!} = (n-p+1) \cdot (n-p+2) \cdots n.$$

Since the numbers $(n-p+1)$ up to $n-1$ are not divisible by p, it follows that the right-hand side is divisible by p^j, but not by p^{j+1}. Hence p^{j+1} does not divide the left-hand side of the equation and this means that $\binom{n}{p}$ is not divisible by p^j. ■

Claim (b). $\binom{n}{p^j} \not\equiv 0 \pmod p$.

Proof. We refine the argument from the previous paragraph. Again we see that

$$(p^j)!\binom{n}{p^j} = (n-p^j+1)\cdot(n-p^j+2)\cdots n.$$

Now we show that the highest power of p that divides the right-hand side also divides $(p^j)!$. Hence, for all $m \in \mathbb{N}$, let $e_p(m) \in \mathbb{N}_0$ denote the exponent of p in the factorization of m into prime powers. This means that $p^{e_p(m)}$ divides m, but $p^{e_p(m)+1}$ does not.

Let $k \in \mathbb{N}$ be such that $k \leq p^j$. Then

$$e_p(k) = e_p(n-p^j+k).$$

(Think about why this is true!) Therefore

$$e_p\big((p^j)!\big) = \sum_{k=1}^{p^j} e_p(k) = \sum_{k=1}^{p^j} e_p(n-p^j+k) = e_p\big((n-p^j+1)\cdots n\big).$$

Now Corollary 1.3.5 forces

$$e_p\left(\binom{n}{p^j}\right) = 0$$

as claimed. ■

Exercise 5.1.10. Here we look for an efficient algorithm that checks whether or not the congruence

$$(P(X))^n \equiv P(X^n) \pmod{n, Q}$$

holds.

Here we have that $n \geq 2$ is a natural number and that Q and P are integer polynomials with coefficients between 0 and $n-1$ and such that $d := \deg P$ is less than $r := \deg Q$. Moreover we suppose that the leading coefficient of Q is coprime to n.

We know from Exercise 3.5.9 that $(P(X))^n \pmod{n, Q}$ can be calculated efficiently and therefore we can efficiently find the (unique) integer polynomial that is congruent to $(P(X))^n$ modulo n and Q, that has degree less than r, and that has coefficients between 0 and $n-1$. Now we write

$$P = a_d X^d + \cdots + a_1 X + a_0.$$

Then

$$P(X^n) = a_d X^{nd} + \cdots + a_1 X^n + a_0.$$

Applying Exercise 3.5.9 once more, we calculate X^{nd}, $X^{n(d-1)}$, ..., X^n efficiently modulo n and Q. For $P(X^n)$ we only need to add $d+1$ polynomials of degree less than r and then we reduce modulo n and Q. Again this can be done efficiently. Finally we compare the coefficients (at most r) modulo n and q and then we know whether the congruence holds or not.

Exercise 5.2.3.

Claim. Let p be a prime number and let P be an integer polynomial. Then for all $m \in \mathbb{N}_0$ we have that

$$(P(X))^{p^m} \equiv P\big(X^{p^m}\big) \pmod{p}.$$

Proof. Here we use the idea that powering with exponent p^m is the same as powering m times with p. Thus we only need to apply Theorem 5.1.1 m times and then we are done. More exactly we argue by induction over m. If $m = 0$, then the claim holds because

$$(P(X))^{p^0} = P(X) = P(X^{p^0}).$$

Now we suppose that the claim is true for some $m \geq 0$. This yields that

$$(P(X))^{p^{m+1}} = \Big(\big(P(X)\big)^{p^m}\Big)^p \equiv \big(P(X^{p^m})\big)^p \pmod{p}.$$

Theorem 5.1.1 implies that moreover

$$\big(P(X^{p^m})\big)^p \equiv P\big((X^p)^{p^m}\big) = P\big(X^{p^{m+1}}\big) \pmod{p},$$

as stated. ■

Exercise 5.2.4. Suppose that n is a composite number that has two distinct prime factors p and q. Moreover let $a \in \mathbb{Z}$ be such that a and n are coprime.

Claim.

$$(X+a)^n \not\equiv X^n + a \pmod{p}.$$

Proof. The proof follows the same idea as in Theorem 5.1.5, but now with Exercise 5.1.8(b).

Let j denote the largest number such that $p^j \mid n$. Then $p^j < n$. The p^j-th coefficient of $(X+a)^n$ is

$$\binom{n}{p^j} a^{p^j},$$

and by Exercise 5.1.8(b) this is not congruent to 0 modulo p. ■

Exercise 6.4.4. Suppose that $n \geq 2$ and that H is a polynomial with leading coefficient coprime to n.

Claim. Let $r \geq 1$ be such that $X^r \equiv 1 \pmod{n, H}$ and let k denote the smallest natural number such that $X^k \equiv 1 \pmod{n, H}$. Then $k \mid r$.

Proof. We argue as in Lemma 3.2.1. First we divide r by k with remainder:

$$r = t \cdot k + r_0,$$

with $t \in \mathbb{Z}$ and $0 \leq r_0 < k$. Then

$$1 \equiv X^r = X^{t \cdot k + r_0} = \left(X^k\right)^t \cdot X^{r_0} \equiv X^{r_0} \pmod{n, H}.$$

Since $r_0 < k$ and by choice of k, we obtain that $r_0 = 0$. Thus $r = t \cdot k$ which means that $k \mid r$, as stated. ■

Bibliography

[AB] Agrawal, M. and Biswas, S.: Primality and identity testing via Chinese remaindering. *Journal of the ACM* **50** (2003), no. 4, 429–443.

[AKS] Agrawal, M., Kayal, N., and Saxena, K.: PRIMES is in P. *Annals of Math.* **160** (2004), no. 2, 781–793.

[AÖ] Ağargün, A. Göksel and Özkan, E. Mehmet: A historical survey of the fundamental theorem of arithmetic. *Historia Math.* **28** (2001), no. 3, 207–214.

[Be] Beutelspacher, A.: *Cryptology.* Mathematical Association of America, 1994.

[Bo] Bornemann, F.: Primes is in P: A breakthrough for "Everyman". *Notices of the AMS* **50** (2003), no. 5, 545–552.

[Br] Bryant, V.: *Yet Another Introduction to Analysis*, Cambridge University Press, 1990.

[Ch] Chen, J. R.: On the representation of a larger even integer as the sum of a prime and the product of at most two primes. *Sci. Sinica* **16** (1973), 157–176.

[Co] Conrey, J. B.: The Riemann Hypothesis. *Notices of the AMS* **50** (2003), no. 3, 341–353.

[CLR] Cormen, T. H., Leiserson, C. E., and Rivest, R. L.: *Introduction to Algorithms.* Third Edition, MIT Press, 2009.

[CM] Coron, J.-S. and May, A.: Deterministic polynomial-time equivalence of computing the RSA secret key and factoring. *J. Cryptology* **20** (2007), 39–50.

[CP] Crandall, R. and Pomerance, C.: *Prime Numbers: A Computational Perspective.* Springer, 2005.

[De] Derbyshire, J.: *Prime Obsession: Bernhard Riemann and the Greatest Unsolved Problem in Mathematics.* Penguin, 2004.

[Dst] Diestel, R.: *Graph Theory.* Fourth edition, Springer, 2010.

[Dtz] Dietzfelbinger, M.: *Primality Testing in Polynomial Time: From Randomized Algorithms to "PRIMES Is in P".* Springer, 2004.

[Do] Doxiadis, A.K.: *Uncle Petros and Goldbach's Conjecture: A Novel of Mathematical Obsession.* Bloomsbury, 2001.

[EFT] Ebbinghaus, H.-D., Flum, J., and Thomas, W.: *Mathematical Logic.* Second edition, Springer, 1994.

[Eb] Ebbinghaus et al.: *Numbers.* Third printing, Springer, 1996.

[Ec] Eccles, P.: *An Introduction to Mathematical Reasoning: Numbers, Sets and Functions.* Cambridge University Press, 1997.

[Frl] Fraleigh, J. B.: *A First Course in Abstract Algebra.* Seventh edition, Pearson, 2003.

[Frz] Franzén, T.: *Gödel's Theorem: An Incomplete Guide to Its Use and Abuse.* Peters, 2005.

[Fü] Fürer, M.: *Faster Integer Multiplication.* Proceedings of the 39th Annual ACM Symposium on Theory of Computing (2007), 57–66.

[G] Granville, A.: It is easy to determine whether a given integer is prime. *Bull. Amer. Math. Soc.* **42** (2005), no. 1, 3–38.

[GT] Green, B. and Tao, T.: The primes contain arbitrarily long arithmetic progressions. *Annals of Math.* (2) **167** (2008), no. 2, 481–547.

[Hal] Halmos, P. R.: *Naive Set Theory.* Van Nostrand, 1960.

[Har] Hardy, G. H.: *A Mathematician's Apology.* Cambridge University Press, 1940.

[HW] Hardy, G. H. and Wright, E. M.: *An Introduction to the Theory of Numbers.* Oxford University Press, 2008.

[He] Helfgott, H. A.: Major arcs for Goldbach's problem, *preprint,* arXiv:1305.2897, 2013.

[Ho] Hofstadter, D.: *Godel, Escher, Bach: An Eternal Golden Braid.* Penguin Books Ltd., 1980.

[HMU] Hopcroft, J. E., Motwani, R., and Ullman, J. D.: *Introduction to Automata Theory, Languages, and Computation.* Pearson/Addison-Wesley, 2007.

[HP] Humphreys, J. F. and Prest, M. Y.: *Numbers, Groups and Codes.* Second edition, Cambridge University Press, 2004.

[J] Jameson, G. J. O.: *The Prime Number Theorem.* Cambridge University Press, 2008.

[Ke] Kelly, T.: The myth of the Skytale. *CRYPTOLOGICA*, Vol. XXII, no. 3 (1998).

[KS] Kurzweil, H. and Stellmacher, B.: *The Theory of Finite Groups.* Springer, 2004.

[LaPe] Laubenbacher, R. and Pengelley, D.: "Voici ce que j'ai trouvé": Sophie Germain's grand plan to solve Fermat's Last Theorem. *Historia Mathematica* **37**, Issue 4, (2010), 641–692.

[LP] Lenstra, H. W. Jr. and Pomerance, C.: Primality testing with Gaussian periods. Preprint, 2005, revised April 2011. `http://www.math.dartmouth.edu/~carlp/aks041411.pdf`.

[LPa] Lewis, H. R. and Papadimitriou, C. H.: *Elements of the Theory of Computation.* Prentice Hall International, 1998.

[LiN] Lidl, R. and Niederreiter, H.: *Finite Fields.* Addison-Wesley, 1983.

[Lo] Lorenz, F.: *Algebra. Volume I: Fields and Galois Theory.* Springer, 2006.

[ME] Murty, M. R. and Esmonde, J.: *Problems in Algebraic Number Theory.* Springer, 2004.

[N] Nair, M.: On Chebyshev-type inequalities for primes. *Amer. Math. Monthly* **89**, no. 2 (1982), 126–129.

[NZM] Niven, I., Zuckerman, H. S., and Montgomery, H. L.: *An Introduction to the Theory of Numbers.* John Wiley & Sons, 1991.

[P] Papadimitriou, C. H.: *Computational Complexity.* Addison-Wesley, 1995.

[RSA] Rivest, R., Shamir, A., and Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. *Comm. of the ACM* **21** (1978), no. 2, 120–126.

[Rob] Robinson, S.: Still guarding secrets after years of attacks, RSA earns accolades for its founders. *SIAM News* **36**, no. 5 (2003).

[Ro] Ross, S.: *A First Course in Probability.* Eighth edition, Pearson, 2008.

[Ros] Ross, P. M.: On Chen's theorem that each large even number has the form $p_1 + p_2$ or $p_1 + p_2p_3$. *J. London Math. Soc.* **10** (1975), 500–506.

[S] Singh, S.: *The Code Book: Science of Secrecy from Ancient Egypt to Quantum Cryptography.* Anchor Books, 2000.

[TZ] Tao, T. and Ziegler, T.: The primes contain arbitrarily long polynomial progressions. *Acta Math.* **201**, no. 2 (2008), 213–305.

[vK] von Koch, H.: Ueber die Riemann'sche Primzahlfunction. *Math. Annalen* **55**, no. 3 (1901), 441–464.

[Za] Zagier, D.: Newman's short proof of the prime number theorem. *American Math. Monthly* **104** (1997), 705-708.

[Zh] Zhang, Y.: Bounded gaps between primes. *Annals of Math.*, to appear.

List of symbols

$\varnothing$	(the empty set), page 8
$\mathbb{C}$	(the complex numbers), page 203
$\mathbb{N}$	(the natural numbers, not including zero), page 7
$\mathbb{N}_0$	(the natural numbers, including zero), page 7
$\mathbb{Q}$	(the rational numbers), page 7
$\mathbb{R}$	(the real numbers), page 7
$\mathbb{Z}$	(the integers), page 7
∞	(infinity), page 8
e	(Euler's constant, $e = 2{,}71828\dots$), page 9
π	(the area of a circle of radius 1, $\pi = 3.14159\dots$), page 48
#M	(number of elements of M), page 8
$N \subseteq M$	(N is a subset of M), page 8
$N \subsetneq M$	(N is a proper subset of M), page 8
$x \in M$	(x is an element of M), page 8
$y \notin M$	(y is not an element of M), page 8
NP	(efficiently verifiable problems), page 69
P	(efficiently computable problems), page 60
RP	(problems with efficient Monte Carlo solutions), page 75
ZPP	(problems with efficient Las Vegas solutions), page 78
$f : N \to M$	(f is a function from N to M), page 9
$f(n) = O(g(n))$	(f grows asymptotically at most as quickly as g), page 59
$:=$	(is defined as), page 8
$a \leq b$	(a is less than or equal to b), page 7
$a < b$	(a is strictly less than b), page 7
$\ln x$	(the natural logarithm of x), page 9
$\log x$	(the logarithm of x to base 2), page 9
$\prod_{i=1}^{n} x_i$	(the product $x_1 \cdot x_2 \cdots x_n$), page 8

$\sum_{i=1}^{n} x_i$	(the sum $x_1 + x_2 + \cdots + x_n$), page 8
$\lfloor x \rfloor$	(the largest integer $n \leq x$), page 9
$\lceil x \rceil$	(the smallest integer $n \geq x$), page 9
$\lvert x \rvert$	(absolute value of x), page 9
$n!$	(n factorial), page 9
$k \mid n$	(k divides n), page 26
$\gcd(a, b)$	(the greatest common divisor of a and b), page 27
$\operatorname{lcm}(a, b)$	(the least common multiple of a and b), page 27
$\operatorname{cp}(n)$	(the set of integers from 1 to $n-1$ that are coprime to n), page 98
$\varphi(n)$	(the number of integers from 1 to $n - 1$ that are coprime to n), page 98
$\pi(n)$	(the number of primes $\leq n$), page 137
$M \bmod n$	(the remainder of M after division by n), page 86
$a \equiv b \pmod{n}$	(a is congruent to b modulo n), page 84
$\operatorname{ord}_n(a)$	(the order of a modulo n), page 94
$\deg P$	(the degree of the polynomial P), page 108
$\deg_n(P)$	(the degree of P modulo n), page 121
$P \equiv Q \pmod{H}$	(P and Q are congruent modulo H), page 113
$P \equiv Q \pmod{n}$	(P and Q are congruent modulo n), page 120
$P \equiv Q \pmod{n, H}$	(P and Q are congruent modulo n and H), page 122
$\mathcal{P}$	(the set of all integer polynomials P satisfying (6.1.2)), page 171
$r(n, k)$	(the smallest prime r with $r \mid n$ or $\operatorname{ord}_r(n) > k$), page 184

Index

How can you tell whether a number is prime? What if the number has hundreds or thousands of digits? This question may seem abstract or irrelevant, but in fact, primality tests are performed every time we make a secure online transaction. In 2002, Agrawal, Kayal, and Saxena answered a long-standing open question in this context by presenting a deterministic test (the AKS algorithm) with polynomial running time that checks whether a number is prime or not. What is more, their methods are essentially elementary, providing us with a unique opportunity to give a complete explanation of a current mathematical breakthrough to a wide audience.

Rempe-Gillen and Waldecker introduce the aspects of number theory, algorithm theory, and cryptography that are relevant for the AKS algorithm and explain in detail why and how this test works. This book is specifically designed to make the reader familiar with the background that is necessary to appreciate the AKS algorithm and begins at a level that is suitable for secondary school students, teachers, and interested amateurs. Throughout the book, the reader becomes involved in the topic by means of numerous exercises.

ISBN: 978-0-8218-9883-3

9 780821 898833

STML/70

For additional information
and updates on this book, visit
www.ams.org/bookpages/stml-70