Chapter 1
Algorithms, An Historical Perspective

Giorgio Ausiello

Abstract The design of algorithms for land measurement, financial transactions and astronomic computations goes back to the third millennium BCE. First examples of algorithms can be found in Mesopotamian tablets and in Egyptians scrolls. An important role in the development of numerical algorithms was played in the ninth century by the Persian mathematician al-Khwarizmi, who introduced the Indian numeration systems to the Arab world and from whom we derived the name 'algorithm' to denote computing procedures. In the Middle Ages algorithms for commercial transactions were widely used, but it was not until the nineteenth century that the problem of characterizing the power of algorithms was addressed. The precise definition of 'algorithm' and of the notion of computability were established by A.M. Turing in the 1930s. His work is also considered the beginning of the history of Computer Science.

1.1 Introduction

The ability to define algorithms for numerical computations or, more generally, as we would say today, for data processing, starts to appear in the history of mankind a few millennia before Christ. Among the most ancient examples of this ability are some tools used for taking note of the results of computations and, especially, the first calendars designed in ancient Egypt. In this chapter, far from attempting a history of algorithms, an effort that would require several volumes by itself, we want to show meaningful examples of algorithms, both numerical and non-numerical, that have been designed, studied and used throughout various historical ages. In the choice and illustration of such examples, there are two most relevant aspects that

G. Ausiello (✉)
Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, via Ariosto 25, 00185 Roma, Italy
e-mail: ausiello@dis.uniroma1.it

G. Ausiello and R. Petreschi (eds.), *The Power of Algorithms*,
DOI 10.1007/978-3-642-39652-6_1,

should be taken into account and that are still important nowadays in the design of modern algorithms. The first one derives from the very notion of algorithm, and corresponds to the need to find the correct sequence of precise and elementary operations that duly executed allow one to reach the solution of a problem in a finite number of steps. The second is related to the need to communicate to other people the sequence of computing steps to be performed and is related, therefore, to the use of a formal and unambiguous language in the presentation of an algorithm.

It is interesting to observe that these two properties (finiteness and formal definability) were understood only recently (less than a century ago, see Sect. 1.7) and are exactly the two properties that nowadays allow us to write, in a suitable programming language, algorithms that can be interpreted and performed by an electronic computing device (see Sect. 2.5). The same properties allow us to highlight the difference between the history of computing and the history of mathematics. It is clear, in fact, that in some sense the history of algorithms is part of the history of mathematics: various fields of mathematics developed due to the need to find solution methods for precise problems.[1] On the other hand are exactly the above-cited finiteness and constructiveness characters that draw a borderline with respect to those fields of mathematics (set theory, function theory, topology, etc.) in which, instead, the study and demonstration of properties of abstract structures have to employ the concepts of the infinitely small and the infinitely large, and often require the use of nonconstructive existential proofs.

1.2 Teaching Algorithms in Ancient Babylonia and Egypt

The oldest nontrivial example of numerical computation that we are aware of is reported on a Sumerian clay tablet from around 2500 BCE, found in Shuruppak, on the Euphrates river. In this example a simple basic problem is addressed, typically related to the life of an agricultural community: the subdivision of the content of a wheat warehouse among various persons in such a way that each person receives a specified amount of wheat. Hence the problem consists in computing how many people can receive their portion of wheat. Actually, the scribe does not present a particular algorithm but just the obtained result. Such a document is nevertheless interesting because we can derive from it information about the number system used by the Sumerians (a mixed decimal and sexagesimal system), and we learn that Sumerians knew various ways to execute division.

More interesting to understanding how algorithms were defined and used in ancient times are some Babylonian tablets from the period 2000 to 1650 BCE. In this case (as in the case of contemporary Egyptian scrolls), the algorithms are presented in a didascalic and repetitive style with reference to specific numerical examples.

[1] For example, Herodotus claimed that the development of geometry in ancient Egypt was due to the need to solve land measurement problems arising from repeated Nile floods.

The number is 4; 10. What is its inverse?
Proceed as follows.
Compute the inverse of 10. You will find 6.
Multiply 6 by 4. You will find 24.
Add 1. You will find 25.
Compute the inverse of 25. You will find 2; 24.
Multiply 2; 24 by 6. You will find 14; 24.
The inverse is 14; 24. This is the way to proceed.

Fig. 1.1 Algorithm for the inversion of the number 4; 10 (that is 250). It is easy to check that 14; 24 is the inverse of 4; 10 since $(4 \times 60 + 10) \times (14 \times 60^{-2} + 24 \times 60^{-3}) = 1$

The number is x. What is its inverse?
Proceed as follows.
[Let y and z be two numbers such that $x = y + z$]
Compute the inverse of y. You will find y'.
Multiply y' by z. You will find t.
Add 1. You will find u.
Compute the inverse of u. You will find u'.
Multiply u' by y'. You will find v.
The inverse is v. This is the way to proceed.

Fig. 1.2 Algorithm for the inversion of the number x. The algorithm is derived from the example in Fig. 1.1 by replacing the numbers appearing in the example with variables

What we can argue is that such tablets were used to teach algorithms to students, that is, to present them with the lists of elementary operations to be performed for each specific numerical example. The general rule (although not explicitly presented) could then be inductively derived from the examples. In the tablets various problems are addressed: square root computations, resolution of second-degree equations, computation of the inverse of a given number, etc. Let's look at one of the examples in detail. This will also offer us the possibility to observe the Babylonian number system more closely.

The problem consists in inverting a given number and, as we said above, the algorithm (Fig. 1.1) does not refer to a generic number denoted by a variable x as we would do today (and as we actually do in Fig. 1.2 in order to illustrate the computation executed by the algorithm), but to a specific value. Let us remember that the inverse of a number x is the number $y = 1/x$ that multiplied by x gives as result the number 1. For example, the inverse of 2 is 30 since choosing a suitable power of the basis 60 we have $2 \times 30 \times 60^{-1} = 1$. Computing the inverse is an important basic operation and, in particular, it was an important operation at that time, since the division operation was performed by multiplying the dividend by the inverse of the divisor. For simple numbers tables of inverses were available. In order to explain the algorithm we have to specify that, in the original text, numbers are represented in cuneiform characters and are expressed in mixed decimal and sexagesimal base. Every number consists of a sequence

of values between 1 and 59, each one expressed in base 10 that here, for the sake of clarity, we present separated by the symbol ";". The number zero simply corresponds to an empty space. The sequence 2; 4; 10, for example, denotes the number $2 \times 60^2 + 4 \times 60 + 10 = 7{,}200 + 240 + 10 = 7{,}450$. Which powers of 60 were used depended on the context, therefore the same sequence might represent the number $2 \times 60^3 + 4 \times 60^2 + 10 \times 60$. In particular, and this has to be regarded as one of the advanced characteristics of Babylonian mathematics, the same notation might be used to express decimal numbers. For example, the above-mentioned sequence 2; 4; 10 might as well represent the number $2 + 4 \times 60^{-1} + 10 \times 60^{-2}$.

In Fig. 1.1 we can see how the hypothetical Babylonian teacher could present the algorithm for the computation of the inverse of the number 4; 10 (corresponding to 250) to his students. The presentation starts as follows: "The number is 4; 10. What is its inverse?" In the same tablet the algorithm is presented several times, each time applied to different numbers expressed in sexagesimal notation. For example, "The number is 8; 20. What is its inverse?", "The number is 1; 13; 20. What is its inverse?", etc.

It is interesting to pay attention to the last sentence in the presentation of the algorithm: "This is the way to proceed." This sentence shows that the person writing the text was conscious of having discovered a computation procedure, in other words an algorithm, to solve the general problem. In fact, the procedure always followed the same steps, independently from the input values. By means of a simple abstraction process it is easy for us to derive the underlying algorithm from the examples (see Fig. 1.2). The computation method is based on the expression $1/x = 1/(y+z) = 1/y \times 1/(z \times 1/y + 1)$ and consists in reducing the computation of the inverse of a given number x to the computation of the inverse of the two smaller numbers y and $u = z \times 1/y + 1$ until we reach numbers for which the inverse is already known (or can be found in precomputed tables of inverses[2]).

A very similar approach in the presentation of algorithms can be found in a famous Egyptian papyrus belonging to the first centuries of the second millennium BCE. The scroll is known as the "Rhind papyrus" from the name of a Scottish traveler who bought some of its fragments, or the "Ahmes papyrus" from the name of the scribe who copied it from an older document. This papyrus is currently held in the British Museum (with the name pBM 10057) and, together with the so-called "Moscow papyrus" and a leather scroll also held in the British Museum, is one of the few documents that provide us with information about the mathematical knowledge in ancient Egypt. Despite its ambitious title, "Accurate reckoning for inquiring into the nature and into the knowledge of all things, all mysteries, all secrets", the document just contains a collection of examples showing how the computation should be carried out in particular cases. Again, as in the case of the Babylonian tablets, the general computation rules (the algorithms) are not explicitly provided in the document, but we can easily infer them from the examples as we did above for the computation of the inverse. The examples provided in the papyrus concern

[2]Babylonians left several tables of simple inverses.

Example of the computation of a triangle of land surface.
If you are told: a triangle is high 10 khet and his base is 4 khet.
What is its area? Do as it has to be done.
Divide 4 by 2. You obtain 2.
Multiply 10 by 2. This is its area.
Its area is 20.

Fig. 1.3 Problem 51 in the Rhind papyrus: algorithm for the computation of the area of a triangle of height 10 and base 4

$$
\begin{array}{lll}
34 \times 1 = 34 & & 21 : 2 = 10 \text{ remainder } 1 \\
34 \times 2 = 34 + 34 = 68 & & 10 : 2 = 5 \text{ remainder } 0 \\
34 \times 4 = 68 + 68 = 136 & & 5 : 2 = 2 \text{ remainder } 1 \\
34 \times 8 = 136 + 136 = 272 & & 2 : 2 = 1 \text{ remainder } 0 \\
34 \times 16 = 272 + 272 = 544 & & 1 : 2 = 0 \text{ remainder } 1
\end{array}
$$

$$34 \text{ x } 21 = 34 \text{ x } (1 + 4 + 16) = 34 + 136 + 544 = 714$$

Fig. 1.4 Multiplication of 34 by 21 with the method of duplicating the first factor and halving the second factor. Note that the sequence 10101 is the binary representation of 21

a large variety of problems: computation of fractions, computation of geometrical series, resolution of simple algebraic equations, and computation of surfaces and volumes.

In order to illustrate the way algorithms are presented in the Rhind papyrus, let us choose a simple example for the computation of the area of a triangle. The sequence of computation steps (somewhat rephrased) is presented in Fig. 1.3.

It is interesting to observe that the style of presentation of the algorithm is quite similar to the style we have found in Babylonian tablets. It is also worth noting that, again in this case, the author is aware of the paradigmatic character of the computation procedure presented in the example when he says: "Do as it has to be done".

Among the algorithms presented in the Rhind papyrus it is particularly relevant to cite the multiplication algorithm based on the so-called technique "by duplicating and halving". The technique is based on the distributive property of multiplication and on the possibility to represent a number as the sum of powers of two (the same property that is behind the binary representation of numbers in computers). Let us consider the following example: Suppose we have to multiply 34 by 21. The same result can be obtained by computing $34 \times (1 + 4 + 16)$ with the advantage that the product of 34 by a power of two can be obtained by means of repeated sums (duplications: see Fig. 1.4). As a consequence, although less efficient than the multiplication algorithm that we use nowadays, the described method does not require knowing the multiplication table (the so-called "table of Pythagoras").

1.3 Euclid's Algorithm

In various beginners' classes in mathematics and computer science, one of the first algorithms that is taught is also one of the most ancient: Euclid's algorithm for the computation of the greatest common divisor of two integer numbers.

The algorithm is presented in book VII of the Elements, Euclid's main work. For over two millennia this book has been a fundamental source for mathematical studies, particularly for geometry and number theory. In the landscape of Greek mathematics, Euclid's algorithm plays a singular role. In fact, in contrast to the kind of mathematics used by Egyptians and Mesopotamian peoples, oriented, as we saw, to the solution of practical problems, Greek mathematics, starting with Thales' work, followed an abstract approach, based on a line of thought that nowadays we would call axiomatic and deductive. On one side, this approach was a big cultural leap and influenced the future course of the discipline, but, on the other side, it put in a secondary place the aspects related to computation and algorithm design. The algorithm for greatest common divisor computation, therefore, is in a sense an exception. At the same time, it has to be noted that the style in which the algorithm is presented by Euclid offers an important step forward with respect to the way in which the computation processes we have seen until now were presented. In Euclid's text, in fact, the algorithm is formulated in abstract terms, with reference to arbitrary values and is not applied to specific integer numbers given as examples. In addition, the algorithm is formulated in geometrical terms. The arbitrary integer values are actually represented by means of segments and the expression "a number measures another number" expresses the idea that the smaller number divides the larger number and can, therefore, be adopted as a unit of measure of the larger number (just as a shorter stick can be used to measure the length of a longer stick).

After defining the concept of relatively prime numbers (segments that have the property that the only segment that divides them both is the unit segment), Euclid proceeds to defining the computation process in the following terms.

> Suppose we are given two numbers AB and $\Gamma\Delta$ that are not relatively prime, and suppose that $\Gamma\Delta$ is the smaller. We have to find the greatest common measure of the numbers AB and $\Gamma\Delta$. If $\Gamma\Delta$ measures AB then $\Gamma\Delta$ is the common measure of AB and $\Gamma\Delta$, since $\Gamma\Delta$ is also a measure of itself. Clearly it is also the greatest common measure since no number larger than $\Gamma\Delta$ can measure $\Gamma\Delta$. In case $\Gamma\Delta$ does not measure AB, if we eliminate $\Gamma\Delta$ from AB a number will exist that measures what is left. The remainder cannot be 1 since otherwise the numbers AB and $\Gamma\Delta$ would have been relatively prime, which is not in the hypothesis. Let us suppose that, measuring AB, $\Gamma\Delta$ leaves the remainder AE smaller than itself and let us also suppose that measuring $\Gamma\Delta$, AE leaves ΓZ smaller than itself. Let us finally suppose that ΓZ measures AE. Then, since ΓZ measures AE and AE measures $Z\Delta$, ΓZ measures $Z\Delta$. But since ΓZ is also a measure of itself, it follows that ΓZ measures the entire $\Gamma\Delta$. Now, since $\Gamma\Delta$ measures BE then ΓZ measures BE and since ΓZ also measures AE, this means that ΓZ measures the entire AB. Hence ΓZ is a common measure of both AB and $\Gamma\Delta$.

The text then goes on to prove that ΓZ is the largest common measure of AB and $\Gamma\Delta$ and finally ends with the statement "*This is what was to be proven*" (Fig. 1.5).

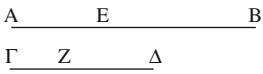

Fig. 1.5 Computation of the greatest common divisor between the length of the segment AB and the length of the segment $\Gamma\Delta$. The result is given by the length of the segment ΓZ

Input: two integer numbers n and m.		
Output: the GCD of n and m.		
Step 1: If $m = n$ then	$GCD(n,m) = n$	
Step 2: else	if $m > n$ then	compute $GCD(m-n,n)$
	else	compute $GCD(n-m,m)$

Fig. 1.6 Euclid's algorithm for the computation of the greatest common divisor (GCD)

In modern terms the algorithm can be more easily formulated in the way it appears in Fig. 1.6.

In the algorithm presented in Fig. 1.6 we use a technique that is widely employed in current programming languages and is now known as recursion. This technique (that as we saw was already implicitly used by Euclid) is inspired by the logical concept of induction (see Chap. 2) and consists in determining the value of a function applied to given arguments (in our case, the function GCD applied to n and m) by making use of the value that the same function would return when applied to smaller arguments ($m - n$ and n or, alternatively, $n - m$ and m).

It is also interesting to observe that Euclid's text provides, at the same time, both the algorithm to be followed for computing the function with arbitrary input values and the proof of its correctness (whose presentation is made easy thanks to the recursion approach used by the algorithm). This appears to be a great step forward, not only with respect to the way the algorithms we saw in the preceding sections were presented, but also with respect to the way algorithms are presented today, often without the support of a rigorous correctness proof. In order to guarantee the correct behavior of computer applications, in fact, it would be appropriate that both the algorithms used and the computer programs that implement them in applications were accompanied by formal proofs of their correctness (see Sect. 2.5). Unfortunately, this only happens rarely, and only for the most sophisticated applications.[3]

[3] In some cases, as users unfortunately realize at their own expense, for economy reasons, programs are written by poorly qualified personnel, without making use of the scientific programming methods that research in computer science has made available. This is why computer programs can sometimes behave differently than they were expected to and can even make errors with disastrous consequences.

1.4 Al-Khwarizmi and the Origin of the Word Algorithm

Few people realize that, when in 772 CE the seat of the caliphate was moved from Damascus to Baghdad, this had an exceptional impact on the history of mathematics and of algorithms. In fact, under the Abassid caliphs, and in particular during the caliphate of al-Mansur, Harun ar-Rashid (the legendary caliph of "One Thousand and One Nights"), al-Mamun and al-Mutasim, Baghdad became a very important center for the development of mathematics and science. The translation into Arabic of Greek scientific works had already started in the sixth and seventh centuries, but the early Islamic period also witnessed various violent actions carried out by fanatic religious people that even led to the destruction of books and other scientific works. During the Abassid caliphate, a more open-minded and rational point of view prevailed. Knowledge was gathered from all regions of the known world, processed and elaborated through a synthetic approach that allowed mathematicians and scientists to realize meaningful progress in all fields.

The last decades of the eighth century and the first decades of the ninth century were a flourishing period for Baghdad from both the economical and the cultural points of view. Not only did revenues from commercial exchanges flow to Baghdad from all regions reached by Arab merchants and ambassadors but also many manuscripts were collected in the many libraries of the city. Around the year 820, al-Mamun founded a scientific academy, the House of Wisdom (Bayt al-Hikma), which consisted of a library and an astronomy observatory where scientists and scholars in all disciplines were invited from abroad.

Among the mathematicians who arrived at the House of Wisdom was the person whose name was given to computational procedures: Abdallah Mohamed Ibn Musa al-Khwarizmi al-Magusi. Although we have limited information about his life, we know that he lived approximately between the years 780 and 850 and that, as his name reveals, he was the son of Musa and was born in Khoresme on the border between the regions that today belong to Iran and Uzbekistan. Recently the Uzbek government dedicated a stamp to him and a statue was placed in what is presumed to be his home town, Khiwa (Fig. 1.7).

It is interesting to observe that, as al-Khwarizmi himself reports, the task assigned to him by al-Mamun upon his arrival in Baghdad was mainly of a practical nature:

> ... to compose a short report concerning computation by means of the rules of restoration and reduction, limited to the simplest and most useful aspects of mathematics that are constantly applied to inheritances, legacies, their sharing out, court decisions and commercial transactions and in all other human business or when land measurements are required, excavations of water channels, geometric computations and similar things.

What al-Khwarizmi really did was of much greater impact: his works were of fundamental relevance in the development of arithmetics and over the centuries have been translated and disseminated, establishing the foundations of medieval mathematical thought.

Several works of al-Khwarizmi reached us through Medieval Latin translations: a treatise on mathematics, one on algebra, one on astronomy, one on geography,

Fig. 1.7 Uzbek stamp representing al-Khwarizmi

and a Hebrew calendar. We suspect that other works have been lost. In particular, al-Khwarizmi's name is related to the introduction of the positional decimal system in Islamic countries (and from there to Christian Europe). As is well known, in such systems, developed first in India, a fundamental role was played by the representation of zero by means of a special symbol, a small circle or a dot. Even if the decimal positional notation was in fact known for a long time (as we have also seen, the Mesopotamians used in a sense a positional system), the an arithmetic treatise by al-Khwarizmi was the first mathematical work to provide a detailed presentation of the rules for executing the four basic operations and for computing with fractions according to such notation. The efficiency of such computing methods, compared to the less efficient methods based on the use of the abacus, determined the dissemination of the Indian numbering system (that indeed we call the "Arabic numbering system") and contributed to making the name of al-Khwarizmi famous and to the use of his name to denote any kind of algorithm.

In addition, al-Khwarizmi's treatise devoted to algebra (*Book of algebra and al-muqabala*) had an important role in the development of mathematical knowledge. In this treatise several algebraic problems are presented, most of which derived from applications (e.g., subdivision of legacies), in particular, a series of algorithms for the solution of first- and second-degree equations with numerical coefficients, duly organized into six different classes according to their structure.

In Fig. 1.8 we provide a simple example of the method used to solve the equation $x^2 + 10x = 39$ (this example would later appear in several medieval algebra textbooks). The method is called "square completion" and, in the particular case of our example, consists in constructing first a square of size x (the unknown value)

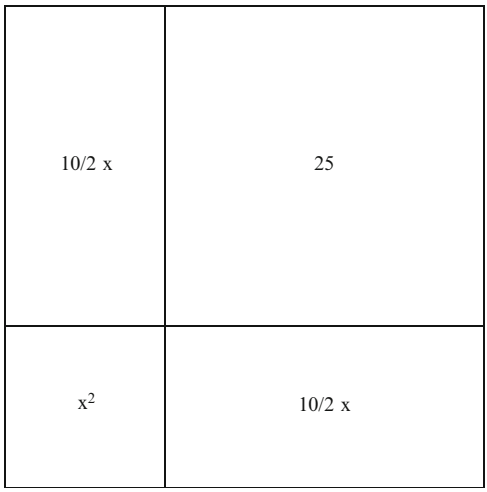

Fig. 1.8 Geometric method for solving the equation $x^2 + 10x = 39$

on whose sides two rectangles of sides x and $10/2$ are built. Finally the larger square is completed by introducing the square with side $10/2$. The larger square that we have constructed in this way has sides of size $x + 5$ and area of size $x^2 + 10x + 25$. But according to what is established by our equation we have: $x^2 + 10x + 25 = 39 + 25 = 64$, and therefore the sides of the larger square have size 8. From $x + 5 = 8$ we may derive $x = 3$.

As we said before, the works of al-Khwarizmi reached us through subsequent Latin versions, and the works of other mathematicians that were inspired by his texts. The number of Latin authors who spread the work of al-Khwarizmi is very large: John of Toledo (*Liber Algorismi de practice arismeticae*), Adelard of Bath (*Liber Ysagogarum Alchorismi in artem astronomicam a magistro A. compositus*), Leonardo Pisano (*Liber abbaci*), Alexander of Villadieu (*Carmen de Algorismo*), John of Halifax, better known as Sacrobosco, (*Algorismus vulgaris*), etc.

It is due to all these authors that the term *Algorismus* eventually became synonymous with computing procedures. For a long time the term was applied only with reference to arithmetical operations, as opposed to computing methods based on the use of abacus. In the *Florence Chronicle* written by Giovanni Villani, for example, we can read that in 1338, in Florence

> we find that boys and girls that learn to read are between eight and ten thousand. The young students that learn the abacus and the algorismus in six schools are between one thousand and one thousand two hundred.

Figure 1.9 shows that, according to an image contained in a book printed in 1508, the contraposition between algorithmic computations and computations based on the use of abacus was still present at the beginning of the sixteenth century.

Only in the eighteenth century did the term 'algorithm' start assuming the broad meaning that it has today. The Encyclopedia of d'Alambert and Diderot provides a definition of the term algorithm as follows:

Fig. 1.9 The contraposition between algorithmic computations and computations based on the use of the abacus, as shown in [96]

Arab term used by some authors and in particular by Spanish authors to identify the practice of Algebra. Sometimes it is also applied to arithmetic operations based on digits. The same word is more generally used to denote method and notation of whatever kind of computation. In this sense we speak of algorithm for integral calculus, algorithm for exponential calculus, algorithm for sine calculus etc.

1.5 Leonardo Fibonacci and Commercial Computing

Among the mathematicians who contributed to spreading al-Khwarizmi's work, a very special role was played by Leonardo Pisano, also known as Fibonacci, who lived between c. 1180 and 1250, whose name is now, for various reasons, famous in the algorithm community (see Fig. 1.10). The main information concerning his origin and his life can be found in his most important work: the *Liber abbaci* that was written in 1202. Fibonacci was the son of a merchant from Pisa who worked in

Fig. 1.10 Leonardo Pisano, also known as Fibonacci

the important warehouse of Bugia (nowadays Béjaïa in Algeria), where he moved in 1192. In this merchant environment Fibonacci learned the algorithms of "Indian arithmetics" ("*Ubi ex mirabilis magisterio in arte per novem figuras Indorum introductus...*") that were extremely effective for commercial computing. Later, as a merchant, he traveled to various places around the Mediterranean (Egypt, Syria, Greece, Sicily and Provence), where he had the chance to upgrade his knowledge of mathematics. In the *Liber abbaci* he explains how he was able to integrate the knowledge of Indian and Arab computing (which he learned from al-Khwarizmi's works) with Euclidean mathematics. Fibonacci was also the author of other treatises such as the *Practica geometriae* and the *Liber quadratorum*. Although, from a mathematical point of view, the last one is probably his most original work, the *Liber abbaci* is undoubtedly the most relevant for its didactic value and for its role in making the name of Fibonacci famous throughout the Western world.[4]

Among the fifteen chapters of the *Liber abbaci* several concern problems of commercial nature; for example, the title of Chap. 9 is "De baractis rerum venalium", and the title of Chap. 10 is "De societatis factis inter consocios". In the volume various practical accounting problems (money exchange, computation of interest, amortization of debts, etc.) are addressed and make this work an important step in the history of accountancy.[5] In any case, the *Liber abbaci* cannot be classified

[4]The name of the volume should not be misunderstood: the book is entirely devoted to the Indo-Arabic computing system and the use of the abacus is never addressed.

[5]Thanks to his competence in this field, in 1241 Fibonacci was in charge of reorganizing the public accounting of the city of Pisa.

Input:	integer n.	
Output:	$F(n)$: nth Fibonacci number.	
Step 1:	If $n = 0$ then	$F(n) = 1$
	Else if $n = 1$ then	$F(n) = 1$
	Else	$F(n) = F(n-1) + F(n-2)$

Fig. 1.11 Algorithm for the computation of Fibonacci numbers

just as an accountancy textbook nor as a handbook of commercial practice (as is the case of other books of the fourteenth and fifteenth centuries). This work is a real mathematical treatise that spans a great variety of topics, from integer arithmetic to fractional computing, from geometry to the solution of algebraic equations, from the computation of arithmetic and geometric series to the calculus of roots of equations.

The 12th chapter of the *Liber abbaci* ("De solutionibus multarum positarum questionum") is a rich source of mathematical problems, especially in the field of recreational mathematics. Among them we can find the famous "rabbit problem" that, as we will see, gained an important role in the history of algorithms: "Quot paria coniculorum in uno anno ex uno pario germinentur". In English the statement of the problem is more or less as follows: a man has a pair of rabbits in a secluded place and we would like to know how many rabbits this pair would generate in 1 year, taking into account that they can generate a pair every month and that after 1 month also the newly born can reproduce.

Fibonacci presents the solution in the following terms:

> Since in the second month the pair generates we will have two pairs in one month. One of these pairs (the first one) generates also in the second month, therefore we will have 3 pairs. After one more month two of them will be fertile and 2 pairs will therefore be born in the third month. We have then 5 pairs. Three of them will then be fertile and hence in the fourth month 3 more pairs will be born and 8 will be the overall number. The last month we will have 377 pairs. You can see in the margin how we operated: we summed the first number with the second then the second with the third, the third with the fourth... in this way you can compute for an infinite number of months.

It is easy to see that the computation procedure can be naturally formulated in recursive terms. The nth value of the sequence is defined as the sum of the $(n-1)$th and the $(n-2)$th values. Usually it is assumed that the value for $n = 0$ and $n = 1$ is 1, and hence the sequence is 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ... In modern terms the sequence (now known as the sequence of Fibonacci numbers) would be defined as in Fig. 1.11.

Beside being defined with the algorithm in Fig. 1.11, Fibonacci numbers can also be expressed in explicit terms by means of the expression: $F(n) = c_1((1 + \sqrt{5})/2)^n + c_2((1 - \sqrt{5})/2)^n$ with suitable coefficients c_1 and c_2 derived from the initial conditions $F(0)$ and $F(1)$.

Fibonacci numbers have several interesting properties. The most important one is that the ratio between $F(n)$ and $F(n-1)$ tends to the constant value $\phi = 1.618$, known as "the mean of Phidias" or the "golden ratio".

Fig. 1.12 Mario Merz, *The Flight of Numbers*, installation

It is worth remembering that the golden ratio is the ratio existing between a segment of unit length and a portion r of it that satisfies the relation $1 : r = r : 1-r$ (in other words, r is defined as the solution of the equation $r^2 + r = 1$). The value $r = 0.618$ (equal to $1/\phi$) is called the "golden section" of the segment.

The golden ratio was used by the ancient Greeks in order to obtain particularly elegant proportions between the dimensions of a building (such as, for example, a temple). More recently the existence of this harmonious ratio between a Fibonacci number and the number that precedes it in the sequence is at the base of the important role that Fibonacci numbers had in architecture for establishing the size of housing modules (Le Corbusier), in music for creating new tonal scales (Stockhausen), and in artistic installations (Merz, see Fig. 1.12).

This very same property is at the root of another reason why the name of Fibonacci is so important in computer science, beside, of course, the role that he played in disseminating Indo-Arabic computation methods. In fact, various data structures (see Fibonacci Heaps in Chap. 4) that allow us to perform operations on large data sets efficiently derive their name from Fibonacci, in particular, those data structures that allow us to handle a set of n records in logarithmic time.[6]

[6]In order to understand the great advantage to using this type of data structures it is enough to observe that searching and updating a database consisting of 500 million records can be done in fewer than 30 steps by making use of Fibonacci trees.

22	47	16	41	10	35	4
5	23	48	17	42	29	29
30	6	24	49	18	12	12
13	31	7	25	43	19	37
38	14	32	1	26	44	20
21	39	8	33	2	27	45
46	15	40	9	34	3	28

Fig. 1.13 Magic square of order 7

1.6 Recreational Algorithms: Between Magic and Games

As we saw in previous sections, throughout the centuries algorithms were developed mostly for practical needs, that is, for solving problems related to the productive activities of humans (agriculture, commerce, construction of buildings). The rabbit problem that is at the origin of the Fibonacci sequence of numbers and of the related algorithm is of a different nature. In this case we see an example that can be classified as recreational algorithmics, in which the design of algorithms is aimed at solving games and puzzles.

Other famous examples of algorithms of this nature are those referring to relational structures (which in the modern language of mathematics and computer science are called graphs), that since the eighteenth century have attracted the attention of mathematicians. As we will see in Chap. 2, such structures were initially defined and studied to solve recreational problems such as the classic problem introduced by Euler, which consisted of deciding whether it was possible to visit all parts of the city of Königsberg going exactly once over all seven bridges that connected them.

In this section we will address two other kinds of problems of a recreational nature that have been studied over the centuries and that stand at the intriguing crossroads between games, religion, and magic: the construction of magic squares and the traversal of labyrinths.

A magic square is a square table containing one integer in each cell arranged in such a way that the result obtained by summing up the integers contained in the cells of each row, of each column, and of each of the two diagonals, is, magically, the same. If the table contains n rows and n columns, the resulting magic square is said to be of "order n". Normally a magic square of order n contains all integers between 1 and n^2, and the magic sum is therefore equal to the sum of the first n^2 integers divided by n, that is $n(n^2 + 1)/2$ (see Induction in Chap. 2). In Fig. 1.13 a magic square of order 7 is shown, where the sum over each row, each column, and each of the two diagonals is equal to 175.

The simplest magic square is of order 3, since no magic square of order 2 can exist. Besides, it is in a sense unique because all the other eight magic squares of order 3 can be obtained by means of symmetries and rotations of the same square.

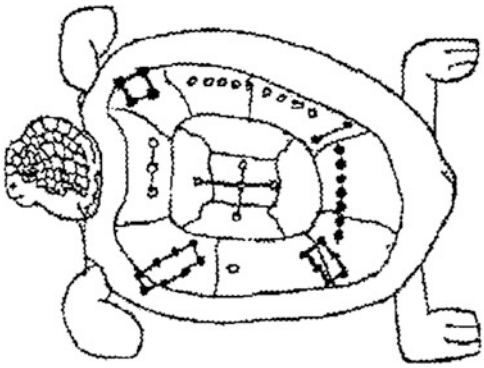

Fig. 1.14 The diagram of the Luo river (China, tenth century)

Such a square is also one of the oldest known magic squares; in fact, versions of this square have been known since the early Chinese Song dynasty (tenth century). The example in Fig. 1.14 is known as the Lo Shu magic square (literally the magic square of the Luo river) and is related to feng shui geomancy. There are 16 magic squares of order 4 and, from them, by means of symmetries and rotations, 880 variants can be obtained. The number of magic squares of order 5 is 275, 305, 224. At present the exact number of magic squares of order 6 is not known.

As we said above, in various historical contexts, magic squares have taken a religious, esoteric, or even curative meaning, and for this reason they were called "magic", probably in the seventeenth century. Still, it is clear that the most interesting properties of such squares are the mathematical properties that since the eleventh century attracted the interest of Arab mathematicians like, for example, al-Haytham. In 1654 Pascal presented a work with the strange title "Treatise on magically magic numbers" to the Paris Academy and, subsequently, in the eighteenth and nineteenth centuries other famous mathematicians such as Fermat, Euler and Cayley analyzed the numerical and algebraic properties of magic squares.

It is not surprising that the interest in magic squares naturally led to the definition of algorithms for their construction. One of the most studied techniques is based on the progressive filling of concentric layers, moving from the outside toward the inside. This technique was formulated for the first time in the thirteenth century by the Arab mathematician al-Zinjani in his "Epistle on the numbers of harmony" and was developed in 1667 by the French scientist Arnauld. We cannot provide details on how the algorithm works here, but it is interesting to quote some comments that the author made on the algorithm. Such comments evidence the attention for the main properties that algorithms should satisfy, simplicity, generality, and efficiency, but also the possibility to demonstrate the correctness of the proposed technique.

> From this I think I can conclude that there does not exist a method which is easier, more synthetic and more perfect for constructing magic squares, one of the most beautiful arithmetical problems. What is peculiar with this method is that figures are written at most twice; we do not proceed by trials but we are always sure that what we do is correct; the largest squares are not more difficult to construct than smaller squares; various solutions can be obtained; nothing is done that cannot be demonstrated.

o			o
	o	o	
	o	o	
o			o

4			1
	7	6	
	11	10	
16			13

4	14	15	1
9	7	6	12
5	11	10	8
16	2	3	13

Fig. 1.15 Construction of a magic square with the marking technique

A second algorithmic technique for constructing magic squares was proposed in the fourteenth century by the Byzantine mathematician Manuel Moschopoulos; it consists of filling a square of odd order n by progressively inserting all integers from 1 to n^2 following suitable rules, similar to the way in which chess pieces are moved on the chess board. Such rules can be expressed formally by making use of modular arithmetic. For example, one of the fundamental rules says that if integer x is in the cell corresponding to row i and column j, if x is not a multiple of n, the number $x + 1$ goes in row $i + 1 \pmod{n}$[7] and column $j + 1 \pmod{n}$, while in the other case (when x is a multiple of n) $x + 1$ goes to row $i + 2 \pmod{n}$ and column j.

The last technique that we want to mention is the one reported by Ibn Qunfudh, in the fourteenth century, in his work *The revelation of operations of computing*. Ibn Qunfudh's method is essentially a marking technique, which is likely to have been introduced centuries before and that can only be applied in the case of squares whose order is a multiple of 4. The method consists in marking, in a suitable way, one half of the cells of a square in such a way that exactly one half of the cells are marked on each row and on each column. Then, starting from the topmost cell on the right (and proceeding row by row from right to left) all integers from 1 to n^2 are progressively enumerated, and those corresponding to the marked cells are inserted in such cells. Finally, starting from the lowest cell on the left, in a similar way, all numbers from 1 to n^2 are again enumerated, and those corresponding to the empty cells are inserted in such cells (Fig. 1.15).

Another class of non-numeric algorithms that were frequently mentioned over the centuries and that have always fascinated experts in recreational mathematics are algorithms for the traversal of labyrinths.

As is well known, just as for magic squares, labyrinths also have taken important meanings and played diverse roles in ancient civilizations. Starting from the most ancient one, the labyrinth of el-Fayum, described by Herodotus, to the mythological labyrinth of the Minotaur, which the archeologist Arthur Evans identified with the Knossos royal palace in Crete, labyrinths have been mostly associated with the symbol of absolute power, the power that it is impossible to reach and, from which,

[7] We have to remember that the value $a \pmod{b}$ corresponds to the remainder of a divided by b; for example $7 \pmod{5} = 2$.

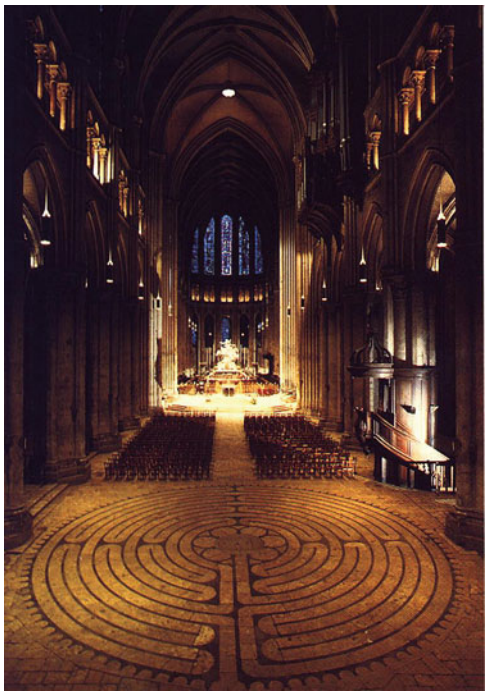

Fig. 1.16 Labyrinth designed on the floor of Chartres cathedral

at the same time, it is impossible to escape. Other meanings have been assigned to labyrinths in religion (see the labyrinths designed on the floor of cathedrals to mean a trail of expiation, as in Fig. 1.16), in magic (labyrinths designed on jewels and amulets), in love (as a metaphor for the pains of the lover), and in architecture (labyrinths of hedges in Renaissance gardens).

We will not devote space here to the description of the various topological structures of labyrinths, nor will we address the issue of labyrinth design, which stimulated several studies across the centuries. We will instead concentrate on the issue of the traversal of labyrinths, either with the aim of reaching the center or of finding the way out.

As far as we know, the first real traversal algorithm was proposed by M. Trémaux in 1892. Technically it is an algorithm for "depth-first" visits, similar to the well-known one based on the idea of following the right wall by always maintaining the right hand on the wall. Different from that one, this algorithm can also be applied in the case where the labyrinth contains circular paths.

In the words of W.H. Matthews, who at the beginning of the twentieth century wrote a book on mazes and labyrinths, Trémaux's algorithm is as follows:

> On arriving at a node which, by the absence of marks, you know you have not already visited, mark the path by which you have just arrived by three marks; if you see by marks on other paths that you have already been to that node, mark the arrival path with one mark only. If now there are no unmarked paths at this node, it means that you have explored

> this particular branch-system and must retrace your steps by the path by which you have arrived. If, however, there are one or more unmarked paths leading from the node, select one of them, and, as you enter it, mark it with two marks. On arrival at a node, you shall never take a path with three marks, unless there are no paths unmarked or with one mark only. When you enter a one-mark path, you shall add two marks so that it is now marked with three marks.

Speaking about Trémaux's algorithm it is nice to remember that the novelist Umberto Eco, in his well-known book *The Name of the Rose*, pretends that the algorithm was already known to the characters of his book in 1327 (the year in which the story is supposed to have happened) but, ironically, he tells us that when Adso of Melk asks "*Does this method allow us to escape from the labyrinth?*", William of Baskerville replies "*Almost never, as far as I know.*"

In conclusion, let us observe that the problem of traversing a labyrinth is still considered one of the fundamental problems in computer science. In fact, it has a paradigmatic value since finding the path between the entrance and the exit of a labyrinth is not very different from finding the connection between two entities in a relational structure, and this is a problem that frequently occurs in several computer applications.

1.7 Algorithms, Reasoning and Computers

Among the various applications of algorithms to human activities, the application to reasoning is certainly, for the reason we will examine in this section, the one that has the most relevant consequences. Automatic reasoning is still, definitely, one of the most ambitious and interesting research domains of artificial intelligence: together with the development of cognitive robotics, the applications of computers to reasoning will have a great impact on the future of mankind. What we want to emphasize here is that even the invention of the first computers derives, although indirectly, from the attempt to transform reasoning into an algorithmic process.

The possibility to realize algorithms that allow one to determine whether a sentence is true or false, and, in such a case, to derive from it all its logical consequences, started to appear in the Western world of philosophy in the seventeenth century, as a consequence of the success of mathematical computing and of the creation of the first computing machines (Pascal's machine is from 1642). Such an ambitious goal arose from the combination of two intellectual dreams that had been cultivated by philosophers from the time of Aristotle and that were reanimated in the Renaissance. The first one was the idea of recording and classifying all universal knowledge, by creating languages and ontologies able to represent all aspects of reality; the second consisted (so simply!) of the search for truth.

The German philosopher Gottfried Leibniz formulated the most ambitious proposal in this direction, based on his interdisciplinary culture, ranging from Aristotelian philosophy to combinatorics and from law to infinitesimal calculus, and animated by an optimistic (almost illuministic) spirit. He thought that it

might be possible to organize all human knowledge in an "encyclopedia" and to apply to knowledge and reasoning the same computation rules that had been successfully applied to mathematics. He felt this could be made possible by using a suitable symbolic system for knowledge representation, which he called *characteristica universalis*. The computation system that he introduced with the name of *calculus ratiocinator* was aimed at extending algebraic methods to logic and thereby to proving or disproving any statement whatsoever. In a nondistant future, Leibniz thought, faced with a problem of any nature, instead of losing time in useless quarrels, "*wise and good willing men*" might sit around a table and say: "*Calculemus!*"; at that point they might run an algorithm (or even turn on a computing machine[8]) that would solve the problem.

The name of Leibniz is often related to the history of computers since he was the first to introduce the binary numbering system and the logical operations on which the binary mathematics used by computers is based. Actually, the relationship between Leibniz and computers is much deeper. We might say that a red line connects directly ideas and works of Leibniz across three centuries with the birth of the first computers.[9]

The construction of the first computers was, indeed, the result of a number of technologically and methodologically convergent developments that took place in the nineteenth and twentieth centuries and of the pressing needs for computing power deriving from the industrial development and (most important) from the military build-up in which the German, British, and American governments were involved from the late 1930s to 1945. This is not the place to illustrate the history of computing machines, but we have to remember that a crucial role in this history (in particular, in the history of the first programmable machines, the real ancestors of today's computers) was played by the studies and works of the English mathematician Alan M. Turing in the mid-1930s. Such studies were devoted, on one side, to formalizing the concept of algorithm and understanding its limits by showing the existence of problems that algorithms are unable to solve. On the other side, Turing's work suggested the possibility to build programmable computers, that is, computers that could execute whatever algorithms were assigned to them in a suitable formal description.

The connection between Leibniz and Turing in computer history, more properly in the history of algorithms, has been established thanks to a problem formulated by another great mathematician, David Hilbert. Among the 23 problems that Hilbert presented at the Mathematics World Congress, held in Paris in 1900, as the main open problems to which the work of mathematicians should be devoted in the

[8] In 1673 Leibniz himself proposed one of the first models of computing machine to the Royal Society: the "*machina arithmetica.*"

[9] It is worth observing that the creation of suitable ontologies allowing us to classify entire domains of human knowledge is still today one of the big challenges of modern computer science. This is related to the aim of providing computer systems (not only those that supervise information search in the Web but also those that execute traditional data management applications) with semantic support that enhances the "intelligence" of software.

twentieth century, the Second Problem emerged as the most important. The problem consisted in establishing whether the axioms of the logical theory of arithmetic were consistent or not, or, in other words, whether it was not possible to derive two contradictory consequences from the axioms.

The issue was more precisely addressed in a famous talk given by Hilbert at the Mathematics World Congress that took place in Bologna in 1928. On that occasion, besides underlining the relevance of the issue of consistency, Hilbert illustrated two other problems concerning the logical theory of arithmetic: the completeness of the theory, that is, the property by which any true assertion should be provable in the theory, and the decision problem (the *Entscheidungsproblem*), that is, the problem of establishing whether there exists an algorithm that, given any logical formula of the theory of arithmetic, is able to tell, in a finite number of steps, whether the formula is valid or not. In other words, Hilbert proposed to address an issue that was much more specific than the ambitious one raised by Leibniz. In fact, the question was not to assess the existence of an algorithm able to decide the truth or falsity of any statement in any field of knowledge but was limited to considering the formalized knowledge in a specific logical domain and to finding out whether an algorithm existed able to decide truth or falsity for an assertion in such a theory. Clearly a positive reply to Hilbert's question would have been encouraging with respect to Leibniz' dream to decide in algorithmic terms any dispute, while a negative answer not only would have implied the end of this dream but would have also indicated a precise limit to the power of algorithmic methods. This type of issue, the conceptual tools needed for addressing them, and the answers that eventually were achieved, had a fundamental role in twentieth century mathematics and also had unforeseen consequences in the development of computer science and of its conceptual basis (see also Chap. 3).

The negative answer to the *Entscheidungsproblem* was given by Alan Turing in 1935–1936. How was this result obtained? In order to show that a problem can be solved by means of an algorithm, it is sufficient to show this algorithm. In this way we can prove the existence of an algorithm to compute the greatest common divisor or to decide whether an integer is a prime number or not. Instead, in order to show that a problem cannot be solved by means of an algorithm (or, as we used to say, to prove that it is an undecidable problem), first of all it is necessary to provide a definition of the concept of algorithm and then to show that algorithms corresponding to this definition cannot solve the given problem. Despite various previous attempts in this direction, in 1935 a satisfactory definition of the concept of algorithm was not known and, hence, first Turing had to find one.

To start with, Turing introduced a notion of algorithm that was based on a very elementary abstract machine model (later to be called "Turing machine"). Such a machine is provided with a (potentially) unlimited tape divided into cells, on which a tape-head (able to move in both directions) can write and read symbols belonging to a suitably defined finite alphabet. In addition, the machine in any moment is in one of a finite set of internal states. For example, a typical behavior rule of the machine could be the following: if the machine is in state q_1 and its head reads the character '2' on the tape, then it should write the character '4' in place of '2' in

the same cell, move the head to the next cell to the right, and enter the state q_2. Note that, in general, a machine that is activated with a particular string x of characters (the "input" string) on the tape can reach a particular "final" state corresponding to the end of the computation. In this case the content of the tape in such a moment can be considered the "result" of the computation, but in some cases it may happen that the machine does not reach the final state and may indefinitely keep working.

The next step of Turing's work was the following: having assumed that the most general kind of algorithm was provided by his machines, he had to identify a problem that could not be solved by these machines, since a problem not solvable by means of Turing machines would be a problem that could not be solved by means of any possible kind of algorithm. Such a problem turned out to be exactly the so-called halting problem of Turing machines. In fact, Turing was able to show that there does not exist any Turing machine able to decide, given a machine M and a string of symbols x, whether M enters a final state or not when it is activated on a tape containing the input string x. Finally, as a last step, Turing showed that the halting problem could be formulated as a particular case of the decision problem stated by Hilbert.

As stated above, Turing reached several results that had a fundamental role in the subsequent developments leading to the construction of the first computers and that are among the most important conceptual pillars of computer science. The first result, clearly, was to provide a definition of the concept of algorithm that, although very simple, has never been substituted by any other definition and still represents the most general notion of algorithm. In fact, no problem is known that can be solved by any algorithm and that cannot be solved by a Turing machine.

The second result was to identify the limits of algorithms. After the proof of the undecidability of the halting problem (and, as a consequence, of the *Entscheidungsproblem*), many more undecidable problems have been discovered in various domains of knowledge (algebra, geometry, coding theory, computer science, etc.). The same limits that hold for Turing machines clearly hold for real computers in the sense that if a problem cannot be solved by means of algorithms, it cannot be solved by any computer, no matter how powerful it is.[10]

The third, fundamental result is closely related to the construction of programmable computers. Although Turing was not thinking in terms of real computing machines when he defined his model, but rather he was trying to represent the way a man's mind proceeds when he performs a computation, it is clear that his notion of algorithm (based on the Turing machine) was leading to the conclusion that any algorithm could be performed with an automatic device. But this might create the impression that we should build a different machine for every problem that we want to solve. The most surprising result that Turing showed

[10]Note that the contrary is not true. As we will see in Sect. 2.3 and in Chap. 3, there are problems that, in principle, can be solved in algorithmic terms but cannot be solved in practice with a computer since their solution may require an amount of time (computation steps) greater than the life of the Universe.

is that, instead, there exist special Turing machines able to simulate the behavior of any other Turing machine. A machine U of this kind, called a *universal Turing machine*, is devised in such a way that if we provide the description of another Turing machine M (the "program" of M) and a sequence of symbols x (the "input" of M) on the tape of U and then we activate U, the universal machine, step by step, executes the rules of the machine M on the string x.

The universal Turing machine had an important role in the invention of programmable computers that in the 1950s replaced the previously built computers, thanks to its greater versatility. A great deal separates those first programmable computers, in which the program had to be manually uploaded in the memory of the computer in the form of a binary string, from the modern computers in which programs are written in high-level languages easily understandable by the users and then translated into binary computer language by suitable software systems running on the computer itself. On the other hand it has been precisely this long history of innovations and technological advances that has made algorithms and computers ubiquitous today.

1.8 Conclusion

In this chapter we have shown, by means of a few paradigmatic examples, how in various historical ages, spanning thousands of years, algorithms were used to solve problems of interest for specific applications or, in some cases, simply to solve recreational problems. The examples also show a variety of styles in which algorithms were presented in different contexts. We have also illustrated how, around the mid-twentieth century, the need to provide a formal definition of the concept of algorithm and the discovery of the existence of universal algorithms (or machines), able to interpret and execute any algorithm presented to them (provided it is written in a suitable formal language) had a crucial role in the construction of the first programmable computers.

Today, thanks to the ubiquitous presence of computers and of computer applications, the concepts of algorithm and of computer program are quite familiar also to non-experts. Nonetheless, it is not as clear to everybody that, thanks to the technological development of computers and of programming languages and to the growing complexity of applications, the design of algorithms has changed from a creative and artisanal activity into a real science. In the next chapter we will show how the design of valid and efficient algorithms requires the use of suitable design techniques and a sophisticated organization of the data to be processed. Then, in the subsequent chapters, we will present examples of algorithms that are present in our everyday life (although hidden in devices such as, for example, our cellular phones) and we will illustrate the advanced design techniques that have been employed in order to obtain from them the required performance and effectiveness.

1.9 Bibliographic Notes

The history of algorithms and the history of mathematics clearly have several points in common. In order to understand such interconnections it is certainly advisable to read a volume devoted to the history of mathematics, such as [10], but, more interesting may be [35], where a study of the role that computing had in the evolution of mathematics is presented.

The history of algorithms is systematically presented in [16]. From this work are taken various examples presented in this chapter. Several volumes illustrate the development of the concept of algorithms in some specific historical periods. Particularly interesting are the books [15] for Egyptian mathematics and [95, 114] for Arab mathematics and, in particular, for the works of al-Khwarizmi. A good illustration of the relevance of the work of Fibonacci in the history of algorithms is contained in [28], while a general historical perspective is provided by Morelli and Tangheroni [81]. The Trémaux algorithm for the traversal of labyrinths is presented in [77].

The interesting relationships among mathematics, philosophy, and esoterism that characterized the search for a language that could be used to classify and represent all human knowledge is well described in [37]. The cultural and conceptual trail that, in the context of Western mathematical thought, leads from Leibniz to Turing and to the invention of programmable computers is illustrated in [23] in a very clear way by one of the greatest contemporary logicians, Martin Davis, who had an important role in the study of undecidability. Readers interested in the life and work of Alan M. Turing can read [60].