ALGORITHM (mediaeval Latin, *algorithmus* from the title of the Persian Mathematician Abu Gafar Muhammad Ibn Musa al-Huwarismi, called al-Chwarizmi, which changed under the influence of Greek to *arithmos*) —the precise description of procedure that guarantees the achievement of a certain result in a finite number of elementary steps.

The most essential features of an algorithm are as follows: (1) the initial situation in which the algorithm may be applied has been defined, as well as the target situation and the kinds of operations that can be performed; (2) the performed elementary operations are completely unambiguous and can be realized by a definite performer of the algorithm. Since it is possible to construct an algorithm intended for different performers, the question of whether a particular set of instructions is an algorithm is a relative question. For some performers the set of instructions is an algorithm, while for others it is not. Sometimes it is required that the performance of particular steps should be purely mechanical, i.e., that no understanding of the operation's meaning should be required; (3) after performing a particular step, the instructions indicate what the following step should be or provide information that the task has been completed. The set of instructions may take the form of conditional instructions: if a particular condition is met, then perform a certain operation; (4) the achievement of the desired result is guaranteed—regardless of the parameters of the initial situation the performance of the algorithm ends with the achievement of the target situation. It may happen, however, that an algorithm cannot be performed practically, since it would require too many means of different kinds (e.g., time, memory, materials); (5) an algorithm is always general and so may be used repeatedly for initial situations with different parameters; (6) the performance of the algorithm is always finite. In this situation, however, we may speak of a desired state of equilibrium and a finite process to achieve this equilibrium.; (7) an algorithm is an abstract object, can be written in many ways, and can be expressed in different languages. The algorithm is thus something distinct from its written expression. Also, the algorithm is distinct from the task it completes.

The development of electronics led to the construction of computers with the result that algorithms became very important: algorithms could be performed automatically in the form of computer programs. A large portion of studies in information science concern algorithms. The following questions are considered: (a) whether an algorithm is correct, especially the proof that a particular algorithm guarantees the solution of the problem presented; (b) estimating the effectiveness of an algorithm; (c) finding the best algorithms for solving problems of a particular type; (d) finding an infallible way to proceed from the description of a problem to an algorithm that solves the problem.

The question of the effectiveness of an algorithm would require a separate discussion. This question is otherwise described as computable complexity, i.e., the amount of time (for elementary operations) and memory needed to realize the algorithm. Classes of algorithms of similar computable complexity are usually described in terms of a Turing machine. The most important of these classes are algorithms that can be performed in polynomial time on a common Turing machine (P-Time) and on an indeterministic Turing machine (NP-Time). These two classes may be understood respectively as those that contain algorithms for which a set of instructions is known that makes it possible to find the desired answer in a specified time as a polynomial of the parameter of the initial data, and those for which there is a way to the result that takes the same time, although we are not necessarily able to indicate the way. The first class is interpreted in terms of time as the class of algorithms that can be realized in practice, while the second class contains most of the other algorithms that resolve problems that are important from a practical point of view. Despite intense research, no one has been able to establish whether these classes overlap. Where a practically executable algorithm cannot be found, other methods are used that do not guarantee that a result will be found (using computer simulations), but which provide practical opportunities (with the help of computer simulations) to find an approximate solution, e.g., evolutionary (genetic) algorithms, or artificial neural networks.

D. E. Knuth, *The Art of Computer Programming* I–III, Reading (Mass.), 1968–1973, 1997–1998^3; A. V. Aho, J. E. Hopcroft, J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Reading (Mass.) 1974(in Polish: *Projektowanie i analiza algoritmów komputerowych*, Wwa 1983); W. M. Turski, *Propedeutyka informatyki* [Propaedeutic of information science], Wwa 1977^2, 1989^3; A. V. Aho, J. E. Hopcroft, J. D. Ullman, *Data Structures and Algorithms*, Reading (Mass.) 1983; D. Harel, *Algorithmics. The Spirit of Computing*, Reading (Mass.) 1987, 1992^2 (in Polish: *Rzecz o istocie informatyki. Algorytmika*, Wwa 1992); P. Dembiński, LFor (passim).

Piotr Kulicki