ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

УДК 519.6

BAAH V. A.
Scientific advisor: ***Stavytskyi O. V.***, PhD
KIBiT, Kyiv

PROBLEMS OF THE THEORY OF COMPUTATIONAL COMPLEXITY

Computer science is the study of the theory, experimentation, and engineering that form the basis for the design and use of computers according to Stanford Encyclopedia of Philosophy [5]. It is the scientific and practical approach to computation and its applications and the systematic study of the feasibility, structure, expression, and mechanization of the methodical procedures (or algorithms) that underlie the acquisition, representation, processing, storage, communication of, and access to, information. An alternate, more succinct definition of computer science is the study of automating algorithmic processes that scale. A computer scientist specializes in the theory of computation and the design of computational systems [5].

Its fields can be divided into a variety of theoretical and practical disciplines. Some fields, such as computational complexity theory (which explores the fundamental properties of computational and intractable problems), are highly abstract, while fields such as computer graphics emphasize real-world visual applications. Other fields still focus on challenges in implementing computation. For example, programming language theory considers various approaches to the description of computation, while the study of computer programming itself investigates various aspects of the use of programming language and complex systems. Interaction between human and computer considers the challenges in making computers and computations useful, usable, and universally accessible to humans. Today we are focusing on the theological face of computer science which is the computational complexity theory.

An important component of any scientific study of a certain set of objects is a classification aimed at identifying the characteristic and general properties of objects. In the field of development of mathematical and algorithmic support of computer systems, such a set of objects are algorithms for solving problems on which the developed software systems are based. Impressive growth in the performance of modern computer systems, obeying the "Moore law" [2], does not reduce the resource requirements for algorithmic support. The composition of these requirements includes the time efficiency of software implementations. This is determined by the fact that a number of today's computational problems have a higher degree of polynomial complexity, which is typical for problems from NP-complete and NP-difficult classes [4]. Another aspect of modern computing problems is a large dimension, for example, in real tasks that are solved using the finite element method, especially inverse problems [1]. The choice of mathematical methods and decision algorithms determines the time characteristics of the software implementation of calculations. The solution of the practically significant problem of choosing a rational computational algorithm can be based on special classification methods.

You can build an algorithm not for any problem. There are algorithmically unsolvable problems. For example the problem of self-applicability of the Turing machine, the problem of stopping the algorithm [3]. Another important example of an algorithmically unsolvable problem is the automatic proof of theorems.

Computational complexity theory is a subfield of theoretical computer science one of whose primary goals is to classify and compare the practical difficulty of solving problems about finite combinatorial objects – e.g. given two natural numbers n and m, are they relatively prime? Given a propositional formula □, does it have a satisfying assignment? If we were to play chess on a board of size n Ч n, does white have a winning strategy from a given initial position? These problems are equally difficult from the standpoint of classical computability theory in the sense that they are all effectively decidable. Yet they still appear to differ significantly in practical difficulty. For having been supplied with a pair of numbers m>n>0, it is possible to determine their relative primality by a method (Euclid's algorithm) which requires a number of steps proportional to log (n). On the other hand, all known methods for solving the latter two problems require a 'brute force' search through a large class of cases which increase at least exponentially in the size of the problem instance [5].

Complexity theory attempts to make such distinctions precise by proposing a formal criterion for what it means for a mathematical problem to be feasibly decidable – i.e. that it can be solved by a conventional Turing machine in a number of steps which is proportional to a polynomial function of the size of its input. The class of problems with this property is known as P – or polynomial time – and includes the first of the three problems described above. P can be formally shown to be distinct from certain other classes such as EXP – or exponential time – which includes the third problem from above. The second problem from above belongs to a complexity class known as NP – or non-deterministic polynomial time – consisting of those problems which can be correctly decided by some computation of a non-deterministic Turing machine in a number of steps which is a polynomial function of the size of its input. A famous conjecture – often regarded as the most fundamental in all of theoretical computer science – states that P is also properly contained in N P– i.e. P " N P [Ibid.].

Demonstrating the non-coincidence of these and other complexity classes remain important open problems in complexity theory. But even in its present state of development, this subject connects many topics in logic, mathematics, and surrounding fields in a manner which bears on the nature and scope of our knowledge of these subjects. Reflection on the foundations of complexity theory is thus of potential significance not only to the philosophy of computer science, but also to philosophy of mathematics and epistemology as well.

Central to the development of computational complexity theory is the notion of a decision problem. Such a problem corresponds to a set X in which we wish to decide membership. For instance the problem PRIMES corresponds to the subset of the natural numbers which are prime – i.e. {n " N#" n is prime}. Decision problems are typically specified in the form of questions about a class of mathematical objects whose positive instances determine the set in question – e.g.

These problems are typical of those studied in complexity theory in two fundamental respects. First, they are all effectively decidable. This is to say that they may all be decided in the 'in principle' sense studied in computability theory – i.e. by an effective procedure which halts in finitely many steps for all inputs. Second, they arise in contexts in which we are interested in solving not only isolated instances of the problem in question, but rather in developing methods which allow it to be efficiently solved on a mass scale – i.e. for all instances in which we might be practically concerned [Ibid.].

The resources involved in carrying out an algorithm to decide an instance of problems can typically be measured in terms of the number of processor cycles (i.e. elementary computational steps) and the amount of memory space (i.e. storage for auxiliary calculations) which are required to return a solution [Ibid.].

The theory of computational complexity is a rapidly developing field of theoretical computer science and covers both purely theoretical issues and issues directly related to practice. Methods of constructing and analyzing effective algorithms are among the most important applications of this theory, as well as modern cryptographic methods. Therefore, familiarity with the basics of complexity theory is certainly useful for anyone who is going to seriously engage in practical programming or theoretical research.

REFERENCES

1. Aleksandrov A.E. Jevoljucionnaja metodologija razrabotki i soprovozhdenija matematicheskogo i programmnogo obespechenija tehnicheskih sistem. M.: Mashinostroenie, 2001.
2. D'jakonov V.P. "Zakon Mura" i komp'juternaja matematika // Exponenta Pro. Matematika v prilozhenijah. 2003. № 1. C. 82–86.
3. Erusalimskij Ja.M. Diskretnaja matematika: Teorija, zadachi, prilozhenija. M: Vuzovskaja kniga, 1998.
4. Gjeri M., Dzhonson D. Vychislitel'nye mashiny i trudnoreshaemye zadachi M.: Mir, 1982.
5. Stanford Encyclopedia of Philosophy [Electronic resource]. – Access mode : https://plato.stanford.edu/entries/computational-complexity/.