IJIEMR Transactions, online available on 11th Mar 2022. Link :http://www.ijiemr.org/downloads.php?vol=Volume-11&issue=ISSUE-02

DOI: 10.48047/IJIEMR/V11/I03/06

Title **DESIGN AND ANALYSIS OF ALGORITHMS.**

Volume 11, Issue 03, Pages: 26-28

Paper Authors

BegimovOybek

[illegible]

International Journal for Innovative Engineering and Management Research

DESIGN AND ANALYSIS OF ALGORITHMS

BegimovOybek
TUIT

ANNOTATION

This article presents all the data on the design and analysis of algorithms. The full information about the algorithms presented in the article and the technology of their development is given.

Keywords: programs, algorithms, instructions, development technology, methods of application

INTRODUCTION

For the first time, one of the most authoritative books on the development and use of algorithms is published in Uzbekistan. Algorithms are the basis of programming that determines how software will use data structures. You will get acquainted with the basic aspects of algorithm construction, basic concepts and definitions, data structures, then move on to the basic methods of algorithm construction, insolvability and methods for solving unsolvable problems, and finally, study randomization in algorithm design. The most complex topics are explained using clear and simple examples, so the book can be used both for self-study by students and by research scientists or computer technology professionals who want to get an idea of the application of certain methods of algorithm design. Algorithmic analysis consists of two fundamental components: the allocation of a mathematically pure core of the problem and the identification of methods for designing a suitable algorithm based on the structure of the problem. And the better the analyst knows the full arsenal of possible design methods, the faster he begins to recognize the "pure" formulations underlying the intricate problems of the real world. Algorithmic problems form the core of computer science, but they rarely appear in the form of neatly packaged, mathematically precise questions. More often they are burdened with a lot of troublesome details tied to a specific case; some of these details are important, others are redundant. As a result, algorithmic analysis consists of two fundamental components: the identification of a mathematically pure core of the problem and the identification of methods for designing a suitable algorithm based on the structure of the task. These two components are interrelated: the better the analyst knows the full arsenal of possible design methods, the faster he begins to recognize the "pure" formulations underlying the intricate problems of the real world.

MATERIALS AND METHODS

Thus, when used with maximum efficiency, algorithmic ideas do not just provide solutions to clearly defined tasks — they form a language for clearly expressing the issues underlying them. The purpose of this book is to convey to the reader this approach to algorithms in the form of a design process that begins with problems encountered across the entire range of computing applications, uses a good understanding of algorithm design methods and the end result of which is the development of effective solutions to such problems. We will study the role of algorithmic ideas in computer science in general and try to link these ideas with a range of precisely formulated tasks for which algorithms are designed and analyzed. In other words, what are the reasons that make us look for solutions to these problems and how do we choose specific ways to formulate them? How do we know which design principles are appropriate in a given situation? Based on the above, we tried to show how to identify "pure" formulations of algorithmic problems in complex computational domains and how to design effective algorithms based on these formulations to solve the problems obtained. To understand a complex algorithm, it is often easiest to reconstruct a sequence of ideas (including unsuccessful approaches and dead ends) that led from the original simplified methods to a solution developed over time. This is how the style of presentation was formed, which does not lead the reader from the formulation of the problem immediately to the algorithm and, in our opinion, better reflects how we and our colleagues approach solving such problems.

General information

The book is intended for students who have completed a two-semester introductory course in computing technologies (standard course "CS1/CS2"), during which they wrote programs for implementing basic algorithms, and working with data structures such as trees and graphs, as well as with basic data structures (arrays, lists, queues and stacks).

RESULTS AND DISCUSSIONS

Since the transition between this course and the first course in algorithms has not yet become standard, the book opens with a presentation of topics that are familiar to students of some educational institutions in the CS1/CS2 course, but in other institutions are included in the curriculum of the first course in algorithms. Accordingly, this part can be considered either as a review or as new material; including it, we hope that the book can be used in a wider range of training courses and with greater flexibility in relation to the initial knowledge required for the reader.

In accordance with the described approach, we develop basic methods for designing algorithms, studying problems from many areas of computer science and related fields. In particular, rather detailed descriptions of possible applications from the field of systems and networks (caching, switching, inter-domain routing on the Internet), artificial intelligence (planning, games, Hopfield networks), pattern recognition (image segmentation), information extraction (intersection point detection, clustering), operations research (air traffic planning) and computational biology (sequence alignment, secondary RNA structure) are given.

The notion of computational insolvability and NP-completeness in particular plays a significant role in the book. This corresponds to our approach to the general algorithm design process. Sometimes an interesting problem that arises in the practical field has an effective solution, and sometimes it turns out to be provably NP-complete; for a full-fledged approach to solving a new algorithmic problem, you must be able to explore both options with equal confidence. Since so many natural problems in computer science are NP-complete, the development of mechanisms for working with unsolvable problems has become a key factor in the study of algorithms, and this topic is widely presented in the book. The conclusion that the problem is NP-complete should be perceived not as the end of the study, but as an invitation to search for approximation algorithms, heuristic local search methods or solvable special cases. Each of these three approaches is extensively discussed in the book.

To simplify the work with such tasks, each chapter includes a section "Exercises with solutions", in which we take one or more tasks and describe in detail the process of finding a solution. For this reason, the discussion of each exercise with a solution turns out to be much longer than a simple recording of a complete, correct solution (in other words, much longer than would be necessary for a full-fledged solution if this task were given to the student for an independent solution). As in the rest of the book, the discussions in these sections should give the reader an idea of the larger processes used to solve problems of this type and ultimately lead to an accurate solution.

It is worth mentioning two more circumstances related to the use of these tasks for independent work in training courses. Firstly, the tasks are ordered approximately in increasing complexity, but this is just an approximate guideline, and we do not want to pay too much attention to it; since the main part of the tasks was developed for students to work independently, large subsets of tasks in each chapter are quite closely related in terms of complexity. Secondly, apart from the initial chapters, you will have to work hard to solve these problems - both in order to link the description of the problem with the algorithmic methods described in the chapter, and for the direct design of the algorithm. In our training course, we assigned about three such tasks for a week.

REFERENCE

- 1.Sedgewick, Robert; Flajolet, Philippe(2013). An Introduction to the Analysis of Algorithms (2nd ed.). Addison-Wesley. ISBN978-0-321-90575-8.
- 2.Greene, Daniel A.; Knuth, Donald E. (1982). Mathematics for the Analysis of Algorithms (Second ed.). Birkhäuser. ISBN3-7643-3102-X.
- 3.Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.&Stein, Clifford (2001). Introduction to Algorithms. Chapter 1: Foundations (Seconded.). Cambridge, MA: MIT Press and McGraw-Hill. pp. 3–122. ISBN0-262-03293-7.
- 5.Sedgewick, Robert (1998). Algorithms in C, Parts 1-4: Fundamentals, Data Structures, Sorting, Searching (3rd ed.). Reading, MA: Addison-Wesley Professional. ISBN978-0-201-31452-6.