Peter Sanders
Kurt Mehlhorn
Martin Dietzfelbinger
Roman Dementiev

Sequential and Parallel
Algorithms and Data Structures

The Basic Toolbox

May 4, 2019

Springer

To all algorithmicists

Preface

An algorithm is a precise and unambiguous recipe for solving a class of problems. If formulated as programs, they can be executed by a machine. Algorithms are at the heart of every nontrivial computer application. Therefore, every computer scientist and every professional programmer should know about the basic algorithmic toolbox: structures that allow efficient organization and retrieval of data, key algorithms for problems on graphs, and generic techniques for modeling, understanding, and solving algorithmic problems. This book is a concise introduction to this basic toolbox, intended for students and professionals familiar with programming and basic mathematical language. Although the presentation is concise, we have included many examples, pictures, informal explanations, and exercises, and some linkage to the real world. The book covers the material typically taught in undergraduate and first-year graduate courses on algorithms.

Most chapters have the same basic structure. We begin by discussing a problem as it occurs in a real-life situation. We illustrate the most important applications and then introduce simple solutions *as informally as possible and as formally as necessary* to really understand the issues at hand. When we move to more advanced and optional issues, this approach gradually leads to a more mathematical treatment, including theorems and proofs. Thus, the book should work for readers with a wide range of mathematical expertise. There are also advanced sections (marked with a *) where we *recommend* that readers should skip them on first reading. Exercises provide additional examples, alternative approaches and opportunities to think about the problems. It is highly recommended to take a look at the exercises even if there is no time to solve them during the first reading. In order to be able to concentrate on ideas rather than programming details, we use pictures, words, and high-level pseudocode to explain our algorithms. A section "Implementation Notes" links these abstract ideas to clean, efficient implementations in real programming languages such as C++ and Java. Each chapter ends with a section on further findings that provides a glimpse at the state of the art, generalizations, and advanced solutions.

The first edition of this book treated only sequential algorithms and data structures. Today, almost every computer, be it a desktop, a notebook, or a smartphone, has multiple cores, and sequential machines have become an exotic species. The reason for this change is that sequential processors have ceased to get proportional performance improvements from increased circuit complexity. Although the number of transistors in an integrated circuit (still) doubles every two years (Moore's law), the only reasonable way to use this transistor budget is to put multiple processor cores on a chip. The consequence is that nowadays every performance-critical application has to be parallelized. Moreover, big data – the explosion of data set sizes in many applications – has produced an enormous demand for algorithms that scale to a large number of processors. *This paradigm shift has profound effects on teaching algorithms.* Parallel algorithms are no longer a specialized topic reserved for a small percentage of students. Rather, every student needs some exposure to parallel algorithms, and parallel solution paradigms need to be taught early on. As a consequence, parallel algorithms should be integrated tightly and early into algorithms courses. We therefore decided to include parallel algorithms in the second edition of the book. Each chapter now has some sections on parallel algorithms. The goals remain the same as for the first edition: a careful balance between simplicity and efficiency, between theory and practice, and between classical results and the forefront of research. We use a slightly different style for the sections on parallel computing. We include concrete programming examples because parallel programming is still more difficult than sequential programming (the programs are available at `github.com/basic-toolbox-sample-code/basic-toolbox-sample-code/`). We also reference original work directly in the text instead of in the section on history because the parallel part is closer to current research.

Algorithmics is a modern and active area of computer science, even at the level of the basic toolbox. We have made sure that we present algorithms in a modern way, including explicitly formulated invariants. We also discuss important further aspects, such as algorithm engineering, memory hierarchies, algorithm libraries, and certifying algorithms.

We have chosen to arrange most of the material by problem domain and not by solution technique. The chapter on optimization techniques is an exception. We find that an organization by problem domain allows a more concise presentation. However, it is also important that readers and students obtain a good grasp of the available techniques. Therefore, we have structured the optimization chapter by techniques, and an extensive index provides cross-references between different applications of the same technique. Bold page numbers in the index indicate the pages where concepts are defined.

This book can be used in multiple ways in teaching. We have used the first edition of the book and a draft version of the second edition in undergraduate and graduate courses on algorithmics. In a first year undergraduate course, we concentrated on the sequential part of the book. In a second algorithms course at an advanced bachelor level or master's level and with students with some experience in parallel programming we made most of the sequential part of the book a prerequisite and concentrated on the more advanced material such as the starred sections and the parts on external memory and parallel algorithms. If a first algorithms course is taught later in the undergraduate curriculum, the book can be used to teach parallel and sequential algorithms in an integrated way. Although we have included some material about parallel programming and several concrete programming examples, the parallel part of the book works best for readers who already have some background in parallel programming. Another approach is to use the book to provide concrete algorithmic content for a parallel programming course that uses another book for the programming part. Last but not least, the book should be useful for independent study and to professionals who have a basic knowledge of algorithms but less experience with parallelism.

Follow us on an exciting tour through the world of algorithms.

Ilmenau, Saarbrücken, Karlsruhe, Heidelberg
August 2018

Martin Dietzfelbinger
Kurt Mehlhorn
Peter Sanders
Roman Dementiev

A first edition of this book was published in 2008. Since then the book has been translated into Chinese, Greek, Japanese, and German. Martin Dietzfelbinger translated the book into German. Actually, he did much more than a translation. He thoroughly revised the book and improved the presentation at many places. He also corrected a number of mistakes. Thus, the book gained through the translation, and we decided to make the German edition the reference for any future editions. It is only natural that we asked Martin to become an author of the German edition and any future editions of the book.

Saarbrücken, Karlsruhe
March 2014

Kurt Mehlhorn
Peter Sanders

Soon after the publication of the German edition, we started working on the revised English edition. We decided to expand the book into the parallel world for the reasons indicated in the preface. Twenty years ago, parallel machines were exotic, nowadays, sequential machines are exotic. However, the parallel world is much more diverse and complex than the sequential world, and therefore algorithm-engineering issues become more important. We concluded that we had to go all the way to implementations and experimental evaluations for some of the parallel algorithms. We invited Roman Dementiev to work with us on the algorithm engineering aspects of parallel computing. Roman received his PhD in 2006 for a thesis on “Algorithm Engineering for Large Data Sets”. He now works for Intel, where he is responsible for performance engineering of a major database system.

Ilmenau, Saarbrücken, Karlsruhe
November 2017

Martin Dietzfelbinger
Kurt Mehlhorn
Peter Sanders