Preface

Background

High-throughput sequencing has recently revolutionized the field of biological sequence analysis, both by stimulating the development of fundamentally new data structures and algorithms, and by changing the routine workflow of biomedical labs. Most key analytical steps now exploit index structures based on the Burrows–Wheeler transform, which have been under active development in theoretical computer science for over ten years. The ability of these structures to scale to very large datasets quickly led to their widespread adoption by the bioinformatics community, and their flexibility continues to spur new applications in genomics, transcriptomics, and metagenomics. Despite their fast and still ongoing development, the key techniques behind these indexes are by now well understood, and they are ready to be taught in graduate-level computer science courses.

This book focuses on the rigorous description of the *fundamental algorithms and data structures* that power modern sequence analysis workflows, ranging from the foundations of biological sequence analysis (like alignments and hidden Markov models) and classical index structures (like k-mer indexes, suffix arrays, and suffix trees), to Burrows–Wheeler indexes and to a number of advanced *omics* applications built on such a basis. The topics and the computational problems are chosen to cover the actual steps of large-scale sequencing projects, including read alignment, variant calling, haplotyping, fragment assembly, alignment-free genome comparison, compression of genome collections and of read sets, transcript prediction, and analysis of metagenomic samples: see Figure 1 for a schematic summary of all the main steps and data structures covered in this book. Although strongly motivated by high-throughput sequencing, many of the algorithms and data structures described in this book are general, and can be applied to a number of other fields that require the processing of massive sets of sequences. Most of the book builds on a coherent, self-contained set of algorithmic techniques and tools, which are gradually introduced, developed, and refined from the basics to more advanced variations.

The book is accompanied by a website

`www.genome-scale.info`

https://doi.org/10.1017/CBO9781139940023.002 Published online by Cambridge University Press

that provides references to implementations of many index structures and algorithms described here. The website also maintains a list of typos and mistakes found, and we encourage the reader to send corrections as requested therein.

This book introduces a number of significant novelties in presenting and organizing its content. First, it raises to a central role the *bidirectional Burrows–Wheeler index*: this powerful data structure is so flexible as to be essentially the only index needed by most sequence analysis primitives, like maximal repeats, maximal unique and exact matches, and alignment-free sequence comparison. In this book we use k-mer indexes, suffix arrays, and suffix trees mostly as *conceptual tools* to help the reader learn the bidirectional Burrows–Wheeler index and formulate problems with its language.

Another key concept that recurs in a large fraction of the book is *minimum-cost network flow*, a flexible model in combinatorial optimization that can be solved in polynomial time. We use this model as a "Swiss Army knife", both by providing a unified presentation of many well-known optimization problems in terms of minimum-cost flow (like maximum-weight bipartite matching and minimum-weight minimum path cover), and by showing that a number of key problems in fragment assembly, transcriptomics, and metagenomics can be elegantly solved by reductions to minimum-cost flow.

Finally, the book spices up the presentation of classical bioinformatics algorithms by including a number of advanced topics (like Myers' bitparallel algorithm), and by presenting inside a unifying framework the dynamic programming concepts that underlie most such algorithms. Specifically, many seemingly unrelated problems in classical bioinformatics can be cast as *shortest-path problems on a directed acyclic graph* (DAG). For example, the book describes the Viterbi algorithm for hidden Markov models as a special case of the Bellman–Ford algorithm, which can itself be interpreted as a solution to the shortest-path problem on a DAG created by layered copies of the input graph. Even the gap-filling problem in fragment assembly is solved through a similar reduction to a DAG problem. The book contains a number of other problems on DAGs, like aligning paths in two labeled DAGs, indexing labeled DAGs using an extension of the Burrows–Wheeler transform, and path covering problems on weighted DAGs arising from alternative splicing.

The book is designed so that key concepts keep reappearing throughout the chapters, stimulating the reader to establish connections between seemingly unrelated problems, algorithms, and data structures, and at the same time giving the reader a feeling of organic unity.

Structure and content

The book adopts the style of theoretical computer science publications: after describing a biological problem, we give precise problem formulations (often visually highlighted in a frame), algorithms, and pseudocode when applicable. Finally, we summarize our results in one or more theorems, stating the time and space complexity of the described algorithms. When we cannot obtain a positive result in the form of a polynomial-time

https://doi.org/10.1017/CBO9781139940023.002 Published online by Cambridge University Press

algorithm, we give an NP-hardness proof of the problem, thereby classifying each problem as either tractable or intractable.

Nonetheless, ad-hoc optimizations and heuristics are important in practice, and we do explain a number of such strategies for many problems. We choose to present these methods inside a special frame, called *insight*. An insight can also contain additional background information, methods explained only by example, mathematical and statistical derivations, and practical bioinformatics issues. Visually separating insights from the main text puts the key algorithmic concepts in the foreground, helping the reader to focus on them and to potentially skip marginal details.

Every chapter ends with a collection of exercises, of varying difficulty. Some exercises ask the reader just to practice the topics introduced in the chapter. Other exercises contain simple, self-contained parts of long proofs, stimulating the reader to take an active part in the derivations described in the main text. Yet other exercises introduce new problem formulations, or alternative strategies to solve the same problems, or they ask the reader to play with variants of the same data structure in order to appreciate its flexibility. This choice makes some exercises quite challenging, but it allows the main text to stay focused on the key concepts. By solving the majority of the exercises, the reader should also gain a broad overview of the field.

A small number of sections describe advanced, often technical concepts that are not central to the main flow of the book, and that can be skipped safely: such sections are marked with an asterisk. The book is designed to be self-contained: apart from basic data structures such as lists and stacks, and apart from basic notions in algorithm complexity, such as the big-oh notation, every chapter of the book builds only on data structures and algorithms that have been described in previous chapters. Therefore, a reader could potentially implement every algorithm we describe, by just reading this book: using the computer science jargon, we could say that the book is entirely "executable". For pedagogical reasons we choose sometimes not to present the most time- or space-efficient algorithm for a problem. In all such cases, we briefly sketch the more efficient variants in the main text, leave them as exercises for the reader, or cite them in the literature section of the chapter.

The book focuses on *algorithm design*. This means that we mainly focus on combinatorial strategies that can be used to solve a variety of different problems, and that we try to find analogies between the solution of every problem and the solution of other problems described in previous chapters. Our focus on design also implies that we do not include in the book any algorithm that requires advanced mathematical tools for analyzing its performance or for proving its correctness: a basic understanding of amortized analysis and of combinatorics is enough to follow the derivations of all worst-case bounds. No average- or expected-case analysis is included in the book, except for a small number of insights that describe algorithms whose worst-case complexity is not interesting.

A significant part of the book focuses on applications of space-efficient data structures. Research on such structures has been very active over the last 20 years, and this field would deserve a textbook on its own. Our goal was not to delve into the technical data structure fundamentals, but to select the minimal setup sufficient to keep the

https://doi.org/10.1017/CBO9781139940023.002 Published online by Cambridge University Press

book self-contained. With this aim, we avoided fully dynamic data structures, entropy-compressed data structures, and data structures to support constant time range minimum queries. These choices resulted in some new insights on algorithm design. Namely, we observed that one can still obtain optimal solutions to a number of suffix tree problems by resorting to amortized analysis on batched queries solvable without the need for any advanced basic data structure. A noteworthy new result in the book exemplifying this sufficiency is that Lempel–Ziv factorization can be obtained space-efficiently in near-linear time with the chosen minimal data structure setup.

Owing to the wide adaptation of the algorithms appearing in the literature to our self-contained minimal data structure setup, we decided to gather all of the references into a literature section appearing at the end of each chapter. This enables an undisturbed text flow and lets us explain the difference between our description and the original work. Some exercises are also directly related to some published work, and in such cases we mention the reference in the literature section. With regard to new results developed for the book or work under preparation, we added some future references in the literature sections.

Finally, almost all algorithms presented in the book are sequential, and are designed to work in random access main memory. By *genome-scale* we mean both a first logical layer of space-efficient and near-linear-time algorithms and data structures that process and filter the *raw data* coming from high-throughput sequencing machines, and a second layer of polynomial-time algorithms that work on the inherently smaller output of the first layer and that solve *semantical problems* closer to biology (for example in transcriptomics and haplotype assembly). The concepts and methodologies detailed in this book are, however, a vantage point for designing secondary-memory algorithms, parallel shared-memory algorithms, GPU algorithms, and distributed algorithms, whose importance is bound to increase in high-throughput sequencing, and in particular in metagenomics. Some exercises explicitly ask the reader to explore such directions, and the last chapter of the book suggestively ends by referring to an existing distributed algorithm, whose sequential version is described in the book.

Target audience

Since the book has a clear focus on algorithmic sequence analysis for high-throughput sequencing, the main audience consists in graduate students in bioinformatics, graduate students in computer science with a strong interest in molecular biology, and bioinformatics practitioners willing to master the algorithmic foundations of the field. For the latter, the insights scattered throughout the book provide a number of techniques that can be of immediate use in practice. The structure of the book is strongly focused on applications, thus the book could be used as an introduction to biological sequence analysis and to high-throughput sequencing for the novice. Selected parts of the book could even be used in an introductory course in bioinformatics; however, such basic topics were not chosen to cover the whole of bioinformatics, but just to give the minimal foundations required to understand the more advanced concepts that appear in later

https://doi.org/10.1017/CBO9781139940023.002 Published online by Cambridge University Press

chapters. Our fresh presentation of recent theoretical topics in succinct data structures, and in biological applications of minimum-cost flow and dynamic programming, might also appeal to the specialist in algorithm design.

Acknowledgements

First and foremost, we wish to warmly thank our families and friends for their constant support throughout the sometimes laborious process of writing a book. We apologize once again for the long nights spent at work, and for frequent periods of pressure and discouragement that they helped us endure. We are also profoundly grateful to the editors, Katrina Halliday and Megan Waddington, for their constant encouragements – particularly important to keep us motivated at the beginning of the writing process. Our deep gratitude goes also to Gonzalo Navarro, Nadia Pisanti, and Romeo Rizzi, for volunteering to read major parts of the book, and for providing invaluable comments on it. Selected chapters were also reviewed by Carlo Comin, Emanuele Giaquinta, Anna Kuosmanen, Paul Medvedev, Martin Milanič, Alberto Policriti, Nicola Prezza, Kristoffer Sahlin, and Leena Salmela. Some chapters were in a quite early phase during such reviews, and we probably introduced new errors during the last weeks of intensive writing, so the reviewers should not be held responsible for any error that slipped through. Veli Mäkinen wishes also to thank the Finnish Cultural Foundation for supporting the sabbatical year that enabled his full concentration on the book.

The selection of content and structure on edit distances, alignments, hidden Markov models, and text indexing was highly influenced by the *String Processing Algorithms*, *Data Compression Techniques*, and *Biological Sequence Analysis* courses taught repeatedly at the Department of Computer Science, University of Helsinki, starting in the nineties with Esko Ukkonen, Jorma Tarhio (now at Aalto University), and Juha Kärkkäinen, and occasionally inherited by other colleagues, including Veli Mäkinen. The seed of this book consisted in a lecture script of some 150 pages by Veli, which gradually emerged in multiple rounds of lectures for the *Biological Sequence Analysis* course at the University of Helsinki, for the *Genome-Scale Algorithmics* course at the University of Bielefeld, and for a tutorial on *Genome-Scale Sequence Analysis Using Compressed Index Structures* at ISMB 2009. The students of the 2015 edition of the Biological Sequence Analysis course gave valuable feedback on a few chapters of the preprint of this book.

However, this initial draft was then heavily rewritten, restructured, and extended, capitalizing on the heterogeneous educational backgrounds of the authors to broaden its original scope. In the end, 11 out of the 16 chapters of this book were written from scratch, while the other 5 were heavily reworked. Some of the new chapters are based on some distinctive expertise that each one of us brought to the book. For example, Djamal brought in competence on succinct data structures, Fabio on genome analysis methods, and Alexandru on graph theory and computational complexity. As unexpected as it might sound, the very experience of writing a book together contributed to establishing a genuine friendship. We also thank our academic advisors, as well

as all our teachers, research colleagues, and students, for their invaluable contribution in enriching our knowledge – as well as our lives. Finally, our understanding of high-throughput sequencing, and of the computational problems that underpin it, had been greatly increased by collaborating with colleagues inside the Finnish Center of Excellence in Cancer Genetics Research, led by Lauri Aaltonen, inside the *Melitaea cinxia* sequencing project, led by Ilkka Hanski, and inside the HIIT bio-focus area on metagenomics, led by Antti Honkela.

Sometimes chance events have long-term consequences. While visiting our research group at the University of Helsinki, Micheaël Vyverman, from Ghent University, gave a talk on maximal exact matches. Veli's lecture script already contained Algorithm 11.3, which uses the bidirectional BWT index to solve this problem, but without any analysis of its time and space complexity. Micheaël's talk inspired us to study this algorithm with more care: soon we realized that its running time is $O(n \log \sigma)$, but the use of a stack caused a space problem. Luckily, Juha Kärkkäinen happened to hear our discussions, and directed us to a classical stack-trick from the quicksort algorithm, which solved our space problem. After even more thinking, we realized that the algorithm to solve maximal exact matches could be adapted to solve a large number of seemingly unrelated problems in sequence analysis, essentially becoming Algorithm 9.3, one of the centerpieces of a book that was still in its infancy. Later we learnt that the related strategy of implicit enumeration of the internal nodes of a suffix tree with bidirectional Burrows–Wheeler index had already been developed by Enno Ohlebusch and his colleagues.

Another random event happened when we were working on a problem related to RNA transcript assembly. We had sent Romeo Rizzi a submitted manuscript containing an NP-hardness proof of a certain problem formulation. After some time, Romeo replied with a short message claiming he had a polynomial-time algorithm for our problem, based on minimum-cost flows. After a stunned period until we managed to get into contact, it of course turned out that we were referring to different problems. The NP-hard formulation is now presented in Exercise 15.11, and the problem Romeo was initially referring to is Problem 15.5. Nonetheless, this initial connection between assembly problems and minimum-cost flow solutions led to the many applications of minimum-cost flow throughout this book.

While we were working hard on this project, Travis Gagie and Simon Puglisi were working even harder to keep the research productivity of the group at a high level. Simon kept the group spirit high and Travis kindly kept Daniel Valenzuela busy with spaced suffix arrays while there was yet another book meeting.

Many students got the opportunity to implement our ideas. These implementations by Jarno Alanko, Jani Rahkola, Melissa Riesner, Ahmed Sobih, and many others can be found at the book webpage.

https://doi.org/10.1017/CBO9781139940023.002 Published online by Cambridge University Press