Preface

The design and analysis of algorithms deals with extracting certain information from instances of computational problems. In a way, this information is "hidden" in the instances, and usually the aim is to come up with a clever way to obtain the desired information with as little effort (for instance, using as little space and time) as possible. In *online computation*, we are facing a situation that somewhat diverges from this setting. Here, the instance is not known in advance, but we gradually get to know it piece by piece over time. However, parts of the final output must already be created before the whole input is known. As an example, consider the *paging problem*, which we will study in the first three chapters of this book. In a nutshell, the task is to manage the *cache* of a computer that can contain a small fraction of the data that is needed during computation. All the data is stored in the larger and much slower main memory, and our goal is to minimize the number of accesses to this slow memory. In other words, the cache should be managed with the goal in mind that the required data is available in the cache as often as possible. The smallest unit of data that can be moved from the main memory to the cache is called a *page*. If a requested page is currently not in the cache, it must be loaded into the cache by replacing another page, since we assume that the CPU can only directly access the cache but not the main memory. This must of course be done *during runtime*, that is, the pages are requested while the operating system is running and executing programs that require different pages in some order. Thus, the basic question that we ask when dealing with this problem is "Which page should be replaced if we do not know anything about the pages that are requested afterwards?"

In this book, we study online computation in different settings. The book it organized as follows. Every chapter begins with an overview of its content and concludes with some bibliographical remarks, historical notes, and literature for further reading. The first three chapters each introduce a different model of computation in an online environment—deterministic, randomized, and with advice. We use the paging problem to serve as an example for each of them.

We start with a very brief description of computing problems and then focus on the concept of computing online. Chapter 1 investigates what can be done in this setting when assuming that the requested data arrives in a worst possible manner. Such a *worst-case analysis* will be used throughout this book. The output of an online algorithm is compared to that of an optimal (offline) algorithm for the given problem; this approach is known as *competitive analysis*. In this context, "optimal" means that the algorithm knows the whole input, and thus has an enormous advantage compared to the *online algorithm* which is analyzed. To model worst-case instances, we introduce an *adversary* that knows the given online algorithm, and constructs the input such that it makes the online algorithm perform as badly as possible compared to an optimal algorithm. Of course, the existence of such an optimal algorithm is merely hypothetical, and it usually obtains a solution quality that cannot be reached when computing online. We illustrate what can be done when dealing with such a situation for the aforementioned paging problem. In particular, we survey different strategies to manage the cache, such as *first in first out*, *last in first out*, *least frequently used*, and *least recently used*. More over, we analyze a broad class of online algorithms for paging called *marking algorithms*.

We study the power of *randomized computation*, that is, algorithms that produce output that is based on randomness, in Chapter 2. In this case, we assume that the adversary still knows the algorithm, but it does not know the outcomes of the random decisions. Here, we compare the expected value of the online algorithm's solution quality to an optimal output on instances which are again constructed by the adversary in a worst-case manner. We will see in Section 2.2 that this allows the design of online algorithms which are exponentially better (in expectation) than any online algorithm that does not use randomness. To show that this result cannot be asymptotically improved, we learn about a lower-bound technique for randomized (online) algorithms, called *Yao's principle*. Moreover, the *ski rental problem* is used as another example in order to compare deterministic and randomized online computation.

A major focus of this book which distinguishes it from other works (and there are very good ones, see Section 1.7) is that we will always try to be very specific about the information which is hidden in a given instance. In online computation, we usually do not care about the running time of the algorithms we construct or study; we are more interested in what we are losing due to computing without knowing the whole input (one might say "the future"). The *advice complexity* of an online algorithm, which is introduced in Chapter 3, measures the amount of information that is needed to obtain a given solution quality. For every online problem we study, we will pose the question of how much we must know about the input to perform well; in a way, this is the *information content* of the problem. Knowing some simple characteristic helps a lot in some cases; in others, full knowledge of the input is required. This additional knowledge is called *advice*, and as a measurement, the advice complexity (in simple words, the size of the advice) tries to gain some insight into what makes an online problem hard. We first investigate how much advice is both necessary and sufficient to design online algorithms for the paging problem

that achieve a certain (for instance, optimal) solution quality. Then we make a general observation that connects online algorithms with advice and randomized online algorithms.

The following five chapters are each devoted to a single online problem, each of which is investigated with respect to the above three models of computation.

Chapter 4 studies the *k-server problem*, which is one of the most prominent online problems. It is of particular interest since, unlike the paging problem, it is not fully understood. This is remarkable as the question "How well can a deterministic or randomized online algorithm perform for the k-server problem?" is unanswered for more than 25 years. We first study some simple subclasses of the problem, and then focus on what can be done when given advice. These results have some interesting implications for randomized online algorithms for the k-server problem.

Chapter 5 deals with a special variant of *online scheduling* where two partially unknown sequences of tasks should be processed in such a way that as much work as possible is parallelized. For this problem, we are mostly interested in comparing randomized online algorithms to those with advice. It turns out that, for this problem, the difference between whether information about the yet unknown parts of the input is supplied or just guessed is rather small.

In Chapter 6, we study an online version of the *knapsack problem*, and our aim is to give a complete picture of what is possible in each of the above three cases. This is done for both the simple and the general version of the problem. In particular, the former has an intriguing behavior when it comes to how much advice is both sufficient and necessary to obtain a certain solution quality. On the one hand, the case of no additional information leads to a situation where any deterministic online algorithm can be forced by the adversary to perform arbitrarily badly. On the other hand, as little as one single bit of advice allows for a solution that is never worse than twice as bad as an optimal one. Any additional bit of advice does not change this until advice is given which has a length that is logarithmic in the input length; this much advice can be used to get a solution that is worse than an optimal one only by a constant factor arbitrarily close to 1. However, to be optimal instead of almost optimal, linear advice is necessary instead of only logarithmic advice.

Chapter 7 studies the *bit guessing problem*, which is a very generic problem that basically captures the essence of what it means to "compute online." Results about deterministic and randomized online algorithms for the problem can be obtained in a rather straightforward manner. However, our main focus is to use results on the hardness of bit guessing to allow statements about the hardness of other online problems by a special kind of reduction. We construct such reductions for three online problems, namely the k-server problem, the online set cover problem, and the disjoint path allocation problem.

Finally, we study different *online graph problems* in Chapter 8. In the online setting considered, the vertices arrive in an online fashion; a vertex is revealed together with all edges that connect it to previously revealed vertices. We start with the coloring problem and present results for both deterministic online algorithms and online algorithms with advice. Last, we investigate an online version of the

minimum spanning tree problem. Here, we are particularly interested in connections between the online and offline versions of the problem when dealing with special graph classes.

This book by no means claims or even tries to be complete in any sense. It should be understood as an introduction to a selected set of topics that are met in online computation. There are, however, many other topics that are not addressed. For a start, there are many different models that are all worth studying but which are not investigated here. There are, for instance, different types of adversaries when dealing with randomization in online computation that do have knowledge about some of the random decisions made by the online algorithms. Furthermore, there are approaches different to that of competitive analysis such as assuming some probability distribution over inputs instead of having an adversary pick one. As a matter of fact, competitive analysis was often criticized for being too pessimistic, which is why a large number of refined models were introduced; this is also one of the reasons why we study the advice complexity as a complementary measurement asking questions that go beyond "How good can an online algorithm be for a given problem?" and that are more along the lines of "What is it that makes a given problem hard?" However, there are also different models of computing with advice that we will not study in detail in this book; they are described in Section 3.6. Last, there are of course many other online problems out there, many of which both pose and answer interesting questions about the power of determinism, randomization, and advice in online settings.

The Audience

This book is intended for computer science students that have some basic knowledge in algorithmics and discrete mathematics; for instance, it is assumed that the reader knows what a binomial coefficient is, how the expected value of a discrete random variable is computed, how to read and apply the Landau symbols (big-$\mathcal{O}$ notation), and how a worst-case analysis of a given approximation algorithm is done. Basically, the reader should be familiar with how theoretical computer scientists see the world. However, most of the ideas and techniques presented in this book are built on basic fundamentals. All in all, a sufficient preparation should been given by an undergraduate course on theoretical computer science, algorithms and data structures, and discrete mathematics.

At any point, the notation is kept as simple and basic as possible, and the intuition behind the proofs is given, and not just the mathematical details. There are, however, some theorems that are stated without a proof. In this case, pointers to these proofs can be found in the section "Historical and Bibliographical Notes" at the end of the corresponding chapter. Sections marked with "$\star$" are technically advanced, but should still be suitable for students on a graduate level.

Another goal of this book is to be useful to researchers who are interested in the concept of advice complexity, and how to apply it. To this end, the basic principles

are discussed in detail, and put into context. Different techniques to obtain lower and upper bounds are described and used, and the reader is challenged to apply her or his knowledge in the exercises. Altogether, there are 101 of them; the solutions are given at the end of this book. The idea behind these exercises is to gain a deeper insight into some details, to try alternative proofs, and, most importantly, to get a good intuition and technical understanding of the results. Most exercises can be answered by readers at an undergraduate level. Exercises that are technically more challenging are again marked with "$\star$."

Acknowledgment

There are many people who increased the quality of this book a lot. First, I would like to thank Juraj Hromkovič for encouraging me to write a book on this topic and for supporting me all the way. Second, I am very grateful to Meike Akveld, Paola Bianchi, Elisabet Burjons, Jérôme Dohrau, Heidi Gebauer, Philipp Hupp, Nils Jansen, Tobias Kohn, Rastislav Královič, Richard Královič, Sacha Krug, Tobias Lieber, Jesper Mikkelsen, Tobias Mömke, Marcel Schöngens, Jasmin Smula, and especially Hans-Joachim Böckenhauer for proofreading parts of the manuscript.

Some chapters of this book are based on parts of my dissertation [97], which I defended in December 2011 at ETH. I had the great pleasure to have Georg Schnitger and Peter Widmayer as co-referees, and I am very thankful for their comments.

Moreover, I had the opportunity to test the material contained in this book in a course "Approximation and Online Algorithms" at ETH, which was so far held in 2013, 2014, 2015, and 2016. The German lecture notes of this class [98] contain a subset of the material that is presented in this book. Some of the students who attended this class contributed a lot to improve this book, especially Tatjana Brülisauer, Mathias Jostock, Sven Hammann, Dominik Müller, and Philipp Schmid.

I am very thankful to Ronan Nugent of Springer for his support.

Last but not least, I want to thank all the people with whom I worked on online algorithms with advice in the last eight years. Without all the discussions we had, this book would not have been possible.

The typesetting of this book was done with LaTeX.

Finally, I hope you enjoy reading this book as much as I enjoyed writing it.

Zürich, September 2016 *Dennis Komm*

http://www.springer.com/978-3-319-42747-8

An Introduction to Online Computation
Determinism, Randomization, Advice
Komm, D.
2016, XV, 349 p. 58 illus., Hardcover
ISBN: 978-3-319-42747-8