SpringerBriefs in Computer Science

Series Editors

Stan Zdonik
Peng Ning
Shashi Shekhar
Jonathan Katz
Xindong Wu
Lakhmi C. Jain
David Padua
Xuemin Shen
Borko Furht
V. S. Subrahmanian
Martial Hebert
Katsushi Ikeuchi
Bruno Siciliano

For further volumes:
http://www.springer.com/series/10028

Michal Forišek · Monika Steinová

Explaining Algorithms Using Metaphors

Springer

Michal Forišek
Department of Computer Science
Comenius University
Bratislava
Slovakia

Monika Steinová
Department of Computer Science
ETH Zürich
Zurich
Switzerland

ISSN 2191-5768 ISSN 2191-5776 (electronic)
ISBN 978-1-4471-5018-3 ISBN 978-1-4471-5019-0 (eBook)
DOI 10.1007/978-1-4471-5019-0
Springer London Heidelberg New York Dordrecht

Library of Congress Control Number: 2013932561

© The Author(s) 2013
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Motivation

> Feynman was a truly great teacher. He prided himself on being able to devise ways to explain even the most profound ideas to beginning students. Once, I said to him, "Dick, explain to me, so that I can understand it, why spin one-half particles obey Fermi-Dirac statistics." Sizing up his audience perfectly, Feynman said, "I'll prepare a freshman lecture on it." But he came back a few days later to say, "I couldn't do it. I couldn't reduce it to the freshman level. That means we don't really understand it."
>
> —David L. Goodstein

The above quotation nicely expresses our feelings about education. It is our opinion that the ability to explain a subject is an indicator of having the deepest understanding of the subject. Without knowing the subject like the back of one's hand, it is impossible to explain it in (relatively) simple terms. This is true for all subjects, but perhaps most significant when the subject is scientific.

In science, the research does sometimes bring a new discovery. The discovery then initiates a wave of new results—both pushing the boundary further and filling in the blanks around the initial discovery. But even once all the blanks have been filled in, it still makes sense to continue doing the research. The goal of this phase is to simplify the results as much as possible: to look for a more systematic approach, to find the most concise way of describing and presenting them, to find the simplest possible proof… In other words, to extract the essence. This phase of research has two major benefits: First, there is the pure scientific benefit of understanding the subject better, which often allows the researchers to discover patterns that were invisible in the first results, or to generalize or otherwise improve the results. The second benefit is for the sake of education. The better we understand what's going on, the easier we can explain it to the next generation of researchers, the faster the research can move forward.

The history of science abounds with examples where a new, better way of looking at things significantly facilitated our progress. For instance, Euclidean geometry was almost without any progress for centuries, until René Descartes came up with the coordinate system. This discovery gave birth to analytic geometry—and suddenly, we saw connections between geometry, algebra, and

number theory. And only with this connection it was later possible to prove that it is not possible to trisect a general angle using just a compass and a straightedge. Later came another significant step: the complex numbers. It turned out that in analytic geometry they can be used to represent angles and rotations easily. And in fact, complex numbers (and the more general quaternions) are still frequently used in modern computer graphics for that very purpose.

The above example is just one of our favorites, but we could have easily picked hundreds of others. They are all consequences of a higher principle: *Our thinking is limited by the language we use*. A more powerful language allows us to be more efficient when dealing with the stuff we already know, and at the same time it helps us grasp and investigate systems that were too complex before.

> Hunagarian mathematician Paul Erdős, although an atheist, spoke of an imaginary book, in which God has written down all the most beautiful mathematical proofs. When Erdős wanted to express particular appreciation of a proof, he would exclaim "This one's from the Book!"
>
> —John Francis

We fully subscribe to this belief. In fact, being computer scientists, we like to push it a bit further: There has to be a second volume to the book—one with all the most beautiful algorithms. Sadly, we also understand why Erdős assigns the authorship to a supernatural entity—writing such a book has to be a task that can never be finished by mere mortals. And our second volume would be even harder to write.

Many researchers (especially mathematicians) view computer science just as one of the many areas of mathematics. We mostly agree with this point of view. However, the field of computer science has some unique features. One of them is very prominent in algorithm design.

In mathematics, there is no such thing as a better or a worse proof. Once you have a proof, the corresponding theorem is proved. Of course, some proofs are short, simple, and illuminating, while others are long, clumsy, filled with special cases, and downright ugly. But a proof is a proof, and when showing that a theorem holds, any proof will do.[1]

But once we move from proving mathematical theorems to designing algorithms, we encounter a whole new dimension: *efficiency*. Not only can an algorithm be nice or ugly, simple or convoluted, we must also take its time complexity into account. Often, finding an algorithm that solves a particular problem is not the end of the road—it may still be possible to find another one that is strictly better.[2]

---

[1] Actually, that's not *entirely* true. For instance, it makes sense to distinguish between constructive and non-constructive proofs.

[2] What does *better* mean? Usually, the asymptotic behavior of the time complexity is what makes an algorithm better. However, there are also many situations in practice where actual execution speed and/or code simplicity matters.

And make no mistake—while there is a strong positive correlation, *beauty* and *efficiency* are still two very different aspects. For instance, consider the Floyd-Warshall algorithm that computes all-pairs shortest paths in a graph:

```
// Floyd-Warshall algorithm:
for i in 1..n:
  for j in 1..n:
    distance[i][j] = edge_length[i][j]   (zero if i=j, inf. if no such edge)
for k in 1..n:
  for i in 1..n:
    for j in 1..n:
      distance[i][j] = min( distance[i][j], distance[i][k] + distance[k][j] )
```

To most of the computer scientists we know, the sheer minimalistic simplicity makes this one of the most beautiful algorithms. Of course, nowadays we already know multiple algorithms that solve the same problem more efficiently—for instance, algorithms that reduce it to special matrix multiplication, and even a randomized $O(n^2 \log n)$ algorithm due to Moffat and Takaoka. But is any of them more beautiful? We leave the decision to you.

Where this Book Comes in

This book describes multiple algorithms. However, it does not aim to be an algorithm textbook.

The book should be readable to a university student interested in algorithms. However, the said students are not our primary audience—their teachers are.

> Teaching is the only major occupation of man for which we have not yet developed tools that make an average person capable of competence and performance. In teaching we rely on the "naturals", the ones who somehow know how to teach.
>
> — Peter F. Drucker

No book will turn a layman into an excellent teacher. But, dear reader, if you are a computer science teacher and you are anything like us, you probably already spent months of your life hunting for resources that would help you be better. It would make us happy if this book turns out to be a helpful resource to you, and helps you a little bit in this life-long quest.

It would be preposterous to claim that our book is an excerpt from "The Book", but we did our best to achieve some kind of a resemblance. Only time will tell how close we made it.

Acknowledgments

Parts of this book were originally published as the authors' research paper titled "Metaphors and Analogies for Teaching Algorithms" on the conference SIGCSE 2012.

This book would not be here at all, were it not for the community of people involved in organizing programming contests in Slovakia. The members of this community are students and faculty members of Comenius University in Bratislava, Slovakia.

The most significant international activity of this community is the preparation of the annual *Internet Problem Solving Contest* (IPSC, http://ipsc.ksp.sk/)—an online competition that tries to push the boundary of, traditional programming contests. A total of 1306 teams from 81 countries have registered for IPSC 2012.

In addition to IPSC, our community is responsible for running the most significant national programming competitions for secondary school students—the Olympiad in Informatics and the Correspondence Seminar in Programming. Twice a year we organize a camp for the best and brightest secondary school students. Most of the materials presented in this book were originally developed for these camps, and later also used in university courses.

We would never be able to get where we are in our lives, were it not for this community. For that, and for all the help with testing and tweaking some of the materials in this book, we are eternally grateful.

Bratislava, Slovakia — Michal Forišek
Zurich, Switzerland — Monika Steinová
December 2012

Contents