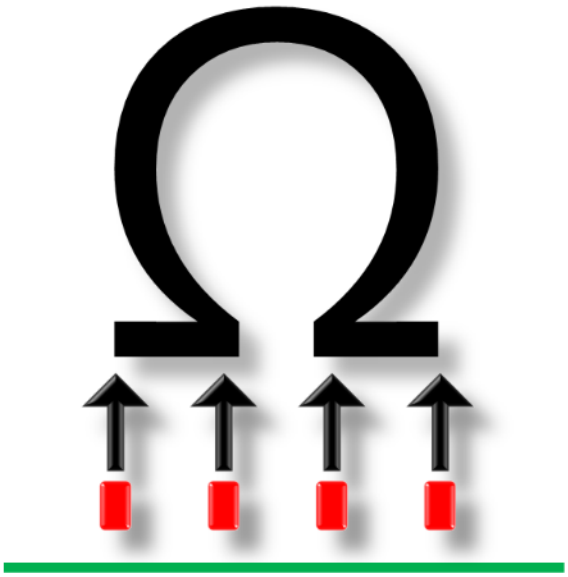

Book 1

Chapter 1-4

9781716745522

This is the 100% identical eBook (PDF) version of CP4 Book 1
that was released on 19 July 2020
Please read https://cpbook.net/errata
for the latest known updates to this PDF

Contents

Forewords for CP4

Bill Poucher

Introduction

In 1970, the Texas A&M UPE Honor Society hosted the first university competitive programming competition in the history of the ICPC. The first Finals was held in 1977 in Atlanta in conjunction with the Winter Meeting of the ACM Computer Science Conference. The ICPC International Collegiate Programming Contest hosted regional competitions at 643 sites in 104 countries for 59 000 team members and their 5043 coaches from over 3400 universities that span the globe. The top 135 teams of three will advance to the ICPC World Finals in Moscow hosted by MIPT scheduled for June 2021.

ICPC alumni number over 400,000 worldwide, many playing key roles in building the global digital community for many decades. The ICPC is the root of competitive programming that reaches out through the global digital community to persons from all cultures and in increasingly-younger generations.

The UVa Online Judge opened the doors for online competition and access to ICPC problems under the direction of Professor Miguel Ángel Revilla. Three of the star-studded team are Steven Halim, Felix Halim, and Suhendry Effendy, authors of Competitive Programming 4, Book 1 and Book 2. Their work will be honored at the ICPC World Finals in Moscow hosted by MIPT with a special award from the ICPC Foundation.

Competitive Programming

What is competitive programming and why should you get involved? First and foremost, it's a mind sport. It more fully develops your algorithmic reasoning skills and bridges the gap between theory and application in bite-sized chunks. Full participation develops problem-solving intuition and competence. Get ready for the Digital Renaissance that will shape your world in the coming decades. To understand the landscape, it is important to shape your mind beyond a swarm of buzzwords. Do it as a team sport.

How do we get started?

Start with Competitive Programming 4, Book 1 and Book 2. Start with Book 1 first :). The authors are seasoned competitive programming experts who have dedicated decades of work to help at all levels of the sport.

In parallel, engage in a culture that develops habits excellence. You are the first generation that has never been disconnected. Being connected is best when we bind our strengths together in common cause. Do that and prepare to meet the challenges that will define your generation.

Life needs you. We are born to compete. We compete best when we compete together, in good faith, in goodwill, and with good deeds. When you come to college, consider the ICPC

and the new program ICPC University Commons that will provide a spectrum of activities that happen outside of the classroom. You can visit `https://icpc.global` for details.

Why get started?

Is developing your problem-solving skills important? Yes. Is preparing for a future engaged in the global digital community important? Yes. Is following T.S. Elliot's advice that to fully develop you must go too far? Yes. Do that in competitive programming. Be careful of pursuits that are not reversible.

Is competitive programming practical? Aristotle asserted that there is nothing more practical than engaging in mental activities and reflections which have their goal in themselves and take pace for their own sake. Let me recommend that you engage your spirit in building a more beautiful world. In the immense scope of life, abundant small kindnesses make a difference. Find friends with common interest and embrace this cycle:

> Repeat for a lifetime: Study; Practice; Rehearse; Dress Rehearse; Perform.

It works for athletes.
It works for musicians.
It works for all performance arts.
It will work for you.

Best, Bill
Dr. William B. "Bill" Poucher, Ph.D., ACM Fellow
Professor of Computer Science, Baylor University
Executive Director, ICPC International Collegiate Programming Contest
President, ICPC Foundation
July 13th, 2020.

Miguel Revilla Rodríguez

Almost 20 years ago (on November 11th, 2003, to be precise), my father (Miguel Ángel Revilla) received an e-mail with the following message:

> "I should say in a simple word that with the UVa Site, you have given birth to a new CIVILIZATION and with the books you write (he meant "Programming Challenges: The Programming Contest Training Manual" [53], coauthored with Steven Skiena), you inspire the soldiers to carry on marching. May you live long to serve the humanity by producing super-human programmers."

What, in my father's words, was *"clearly an exaggeration"*, caused some thinking. And it's not a secret that thoughts can easily lead to dreams. His dream was to create a community around the project he had started, as part of is teaching job at the University of Valladolid, Spain, that gathered people from all around the world working together towards the same ideal, the same quest. With a little searching, on the primitive Internet of the first years of our century, a whole online community of excellent users and tools, built around the UVa site, came to light.

The website *Methods to Solve*[1], created by a very young student from Indonesia, was one of the most impressive among them. There was the result of the hard work of a real genius of algorithms and computer science. The seed was planted to believe that the dream could come true. Moreover, it was not only that the *leaves* of that growing tree were a perfect match, but the root of both projects were exactly the same: to serve the humanity. That young student, the author of the e-mail and the website that put my father to dream, was Steven Halim. Later he would discover that Steven was not alone in his quest, as his younger brother, Felix, shared his view, his interests, and his extraordinary capabilities.

After 15 years of fruitful collaboration and, more important, friendship with Steven and Felix, my father sadly passed away in 2018. His work, and his dreams, now belong to us, the next generation. This book is the living proof that the dream has become true.

"I can't imagine a better complement for the UVa Online Judge", are my father's words. Now, with this fourth version of *Competitive Programming* in my hands, I can add that I can't imagine the very existence of the Online Judge without this book. Both projects have grown in parallel and are, no doubt, perfect complements and companions to each other. By practicing and mastering most programming exercises in this book, the reader can learn how to solve hundreds of tasks and find a place in the top 500 best Online Judge coders. You have in your hands over 2000 (yes, two thousand!) selected, classified, and carefully commented problems from the Online Judge.

The authors, in the past two decades, have grown from contestants, to coaches and, finally, masters in the art of competitive programming. They perfectly know every curve and crossroad in that long path, and they can put themselves in the skins of the young IOI contestant, the ICPC newcomer or the seasoned coach, speaking to each in their own language. This book is, for that very reason, the perfect reading for all of them. No matter if you are starting as a competitive programmer in your local IOI, or are coaching in the next ICPC World Finals, no doubt this IS the book for you.

[1] Please visit `https://cpbook.net/methodstosolve`

I love movies, I adore classic movies, and I know that what I'm watching is a masterpiece, when, after the film ends, I can't wait to start all over again. In Steven and Felix own words *"the book is not meant to be read once, but several times"*. And you will find that same feeling, not only because the authors recommend it, but because you will be anxious to read and re-read it as, like in the greatest movies, you will find something new and amazing each time. This book is, by that logic, a masterpiece.

I also have the great honor of being the Spanish language translator of this book. Translating requires a very meticulous process of converting the words while keeping the spirit. You have to think as the author would think, and have to perfectly understand not only what the author is saying, but also what the author is meaning. It is a handcrafting exercise. Having gone forth and back through this text hundreds of times, I have enjoyed every concept, every new idea, and every tip, not only by what is written in it, but also by what it wants to achieve. The quest of making better programmers and, behind that, the quest of serving humanity. This book is, indeed, a truly masterpiece.

Once you've read this book several times, you will realize how much a better programmer you are but, believe it or not, you will realize that you are also a happier person.

Miguel Revilla Rodríguez (Miguel Jr)
Online Judge Manager
`https://onlinejudge.org`
July 1st, 2020, Valladolid.

Fredrik Niemelä

I got my first physical copy of this book from Steven at IOI 2012 in Italy. Like so many other computer scientists, he has a great sense of humor, and named it "Competitive Programming: Increasing the Lower Bound of Programming Contests." It was the second edition of the book and already twice the size of the first edition. Packed with practical advice, it was well-suited to get beginners started and had useful material for the more seasoned algorithmist.

Steven and Felix's vision for their book was to teach everybody how to program (As Gusteau from Ratatouille would put it: "Tout le monde peut programmer"). I had a similar vision, but instead of writing a book, we created Kattis. "Competitive Programming" and Kattis share this motivating principle: to make learning computer science and programming accessible for everyone. In that sense, they are like two of many pieces in the same puzzle.

Kattis is an online tool for teaching computer science and programming, which relies on a curated library of programming tasks. I managed to convince Steven that he should try using Kattis for some of his teaching activities. Over the years he has moved from using Kattis, to pushing us to improve Kattis, to adding high-quality content to Kattis.

From years of teaching algorithms and using similar systems that preceded Kattis, we learned that the quality of the problems, and their absolute correctness, are paramount for learning outcomes. So, this is where we put extra effort into Kattis. (If you ever felt that it's too much work to add problems to Kattis, this is why). What we did back then is now standard practice—both the ICPC and IOI use the same kinds of methods for their finals.

In this fourth edition (more than twice as large as the second edition!), Steven and Felix, now joined by co-author Suhendry, are using problems from Kattis. We are honored to be included. Finally, our puzzle pieces are directly connected, and I am very excited about that.

I hope you will find this book informative and helpful and that you will spend the time it asks of you. You will not be disappointed.

Fredrik Niemelä
Founder of Kattis
ICPC Contest System Director
IOI Technical Committee Founding Member
`https://www.kattis.com`
July 11th, 2020.

Brian Christopher Dean

I've had the privilege to be part of the competitive programming world for more than three decades, during which time I've seen the field grow substantially in terms of its impact on modern computing. As director of the USA Computing Olympiad and coach of my University's ICPC teams, I have seen firsthand how competitive programming has become a key part of the global computing talent pipeline - both academia and industry are now filled with present-day superstars who were formerly superstars in competitive programming.

Just as the world of competitive programming has shown tremendous growth in scope, depth, and relevance, so too has this text, now in its fourth edition. Earlier editions of this book provided what I consider to be the gold standard for both an introduction and a thorough reference to the algorithmic concepts most prevalent in competitive programming. The same remains true for this edition.

Competitive programming can be a daunting undertaking for the novice student - learning to code is plenty challenging by itself, and on top of this we add a layer of "standard" algorithms and data structures and then another layer of problem-solving insight and tricks. This text helps the introductory student navigate these challenges in several ways, by its thoughtful organization, extensive practice exercises, and by articulating ideas both in clear prose and code. Competitive programming can also be a daunting prospect for the advanced student due to its rapid pace of evolution - techniques can go from cutting-edge to commonplace in a matter of just a few years, and one must demonstrate not only proficiency but true mastery of a formidable and ever-expanding body of algorithmic knowledge. With its comprehensive algorithmic coverage and its extensive listing of ≈ 3458 categorized problems, this text provides the advanced student with years of structured practice that will lead to a high baseline skill level.

I think this is a book that belongs in the library of anyone serious about computing, not just those training for their first or their hundredth programming competition. Ideas from competitive programming can help one develop valuable skills and insight - both in theory and implementation - that can be brought to bear on a wide range of modern computing problems of great importance in practice. Algorithmic problem solving is, after all, truly the heart and soul of computer science! These types of problems are often used in job interviews for a good reason, since they indicate the type of prospective employee who has a skill set that is broadly applicable and that can adapt gracefully to changes in underlying technologies and standards. Studying the concepts in this text is an excellent way to sharpen your skills at problem solving and coding, irrespective of whether you intend to use them in competition or in your other computational pursuits.

I've thoroughly enjoyed reading successive drafts of this updated work shared with me by the authors at recent IOIs, and I commend the authors on the impressive degree to which they have been able extend the scope, clarity, and depth of an already-remarkable text.

Brian Christopher Dean
Professor and Chair
Division of Computer Science, School of Computing
Clemson University, Clemson, SC, USA
Director, USA Computing Olympiad
July 5th, 2020
`http://www.usaco.org/`

Testimonials of CP1/2/3

"Competitive Programming 3 has contributed immensely to my understanding of data structures & algorithms. Steven & Felix have created an incredible book that thoroughly covers every aspect of competitive programming, and have included plenty of practice problems to make sure each topic sinks in. Practicing with CP3 has helped me nail job interviews at Google, and I can't thank Steven & Felix enough!"
— *Troy Purvis, Software Engineer @ Google.*

"Steven and Felix are passionate about competitive programming. Just as importantly, they are passionate about helping students become better programmers. CP3 is the result: a dauntless dive into the data structures, algorithms, tips, and secrets used by competitive programmers around the world. Yet, when the dust settles on the book, the strongest sillage is likely to be one of confidence—that, yes, this stuff is challenging, but that you can do it." — *Dr. Daniel Zingaro, Associate Professor Teaching Stream, University of Toronto Mississauga.*

"CP-Book helped us to train many generations of ICPC and IOI participants for Bolivia. It's the best source to start and reach a good level to be a competitive programmer." — *Jhonatan Castro, ICPC coach and Bolivia IOI Team coach, Universidad Mayor de San Andrés, La Paz, Bolivia.*

"Reading CP3 has been a major contributor to my growth, not just as a competitive programmer, but also as a computer scientist. My entire approach to problem solving has been improved by doing the exercises in the book; my passion for the art of problem solving, especially in contest environments, has been intensified. I now mentor several students using this book as a guide. It is an invaluable resource to anyone who wants to be a better problem solver." — *Ryan Austin Fernandez, Assistant Professor, De La Salle University, Manila, Philippines.*

"I rediscovered CP3 book on 2017-2019 when I come back to Peru after my master in Brazil, I enjoyed, learned and solved many problems, more than during my undergraduate, coaching and learning together in small group of new students that are interesting in competitive programming. It kept me in a constantly competition with them, at the end they have solved more problems than me." — *Luciano Arnaldo Romero Calla, PhD Student, University of Zurich.*

"CP1 helped my preparation during national team training and selection for participating the IOI. When I took the competitive programming course in NUS, CP2 book is extensively used for practice and homework. The good balance between the programming and theoretic exercises for deeper understanding in the book makes CP book a great book to be used for course references, as well as for individual learning. Even at the top competitive programming level, experts can still learn topics they have not learnt before thanks to the rare miscellaneous topics at the end of the book."
— *Jonathan Irvin Gunawan, Software Engineer, Google.*

"Dr. Steven Halim is one of the best professors I have had in NUS. His intuitive visualizations and clear explanations of highly complex algorithms make it significantly easier for us to grasp difficult concepts. Even though I was never fully into Competitive Programming, his book and his teaching were vital in helping me in job interviews and making me a better coder. Highly recommend CP4 to anyone looking to impress in software engineering job interviews." — *Patrick Cho, Machine Learning Scientist, Tesla.*

"Flunked really hard at IOI 2017, missing medal cutoff by 1 place. Then at the beginning of 2018 Steven Halim gave me a draft copy of CP3.1 / CP4 and I ended up getting a gold medal!" — *Joey Yu, Student, University of Waterloo, SWE Intern at Rippling, IOI 2018 Gold Medalist.*

"As a novice self-learner, CP-book helped me to learn the topics in both fun and challenging ways. As an avid and experienced CP-er, CP-book helped me to find a plentiful and diverse problems. As a trainer, CP-book helped me to plan ahead the materials and tactical strategies or tricks in competition for the students. As the person ever in those three different levels, I must effortlessly say CP-book is a must-have to being a CP master!" — *Ammar Fathin Sabili, PhD Student, National University of Singapore.*

"I've been in CP for three years. A rookie number for all the competitive programmers out there. I have a friend (still chatting with him today) who introduced me to this book. He's my roommate on our National Training Camp for IOI 2018's selection. I finally get a grab of this book in early 2019. To be honest I'm not the 'Adhoc' and good at 'Math' type of CP-er. I love data structures, graph (especially trees) And this CP3 book. Is a leap of knowledge. No joke. I met Dr Felix when I was training in BINUS, I also met Dr Steven when I competed in Singapore's NOI and one of my unforgettable moment is, this legend book got signed by its two authors. Even tho the book is full of marks and stains, truly one of my favorite. Kudos for taking me to this point of my life."
— *Hocky Yudhiono, Student, University of Indonesia.*

"I bought CP3 on 7th April 2014 on my birthday as a gift for myself and it has been the most worth-it 30USD spent by me on any educational material. In the later years, I was able to compete in IOI and ICPC WF. I think CP3 played a very big factor in igniting the interest and providing a strong technical foundation about all the essential topics required in CP." — *Sidhant Bansal, Student, National University of Singapore.*

"I have always wanted to get involved in competitive programming, but I didn't know how and where to get started. I was introduced to this book while taking Steven's companion course (CS3233) in NUS as an exchange student, and I found the book to be really helpful in helping me to learn competitive programming. It comes with a set of Kattis exercises as well. This book provides a structured content for competitive programming, and can be really useful to anyone ranging from beginners to experts. Just like CLRS for algorithms, CP is THE book for competitive programming." — *Jay Ching Lim, Student, University of Waterloo.*

"My memories about CP3 is me reading it in many places, the bus, my room, the library, the contest floor...not much time had passed since I start in competitive programming reading CP3 until I got qualified to an ICPC World Final."
— *Javier Eduardo Ojeda Jorge, ICPC World Finalist, Universidad Mayor de San Simón, Software Engineer at dParadig, Chile*

Preface for CP4

This Competitive Programming book, 4th edition (CP4) is a must have for every competitive programmer. Mastering the contents of this book is a necessary (but admittedly not sufficient) condition if one wishes to take a leap forward from being just another ordinary coder to being among one of the world's finest competitive programmers.

Typical readers of Book 1 (only) of CP4 would include:

1. Secondary or High School Students who are competing in the annual International Olympiad in Informatics (IOI) [31] (including the National or Provincial Olympiads) as Book 1 covers most of the current IOI Syllabus [16],
2. Casual University students who are using this book as supplementary material for typical Data Structures and Algorithms courses,
3. Anyone who wants to prepare for typical fundamental data structure/algorithm part of a job interview at top IT companies.

Typical readers of **both** Book 1 + Book 2 of CP4 would include:

1. University students who are competing in the annual International Collegiate Programming Contest (ICPC) [57] Regional Contests (including the World Finals) as Book 2 covers much more Computer Science topics that have appeared in the ICPCs,
2. Teachers or Coaches who are looking for comprehensive training materials [21],
3. Anyone who loves solving problems through computer programs. There are numerous programming contests for those who are no longer eligible for ICPC, including Google CodeJam, Facebook Hacker Cup, TopCoder Open, CodeForces contest, Internet Problem Solving Contest (IPSC), etc.

Prerequisites

This book is *not* written for novice programmers so that we can write much more about Competitive Programming instead of repeating the basic programming methodology concepts that are widely available in other Computer Science textbooks. This book is aimed at readers who have at least basic knowledge in programming methodology, are familiar with at least one of these programming languages (C/C++, Java, Python, or OCaml) but preferably more than one programming language, have passed (or currently taking) a basic data structures and algorithms course and a discrete mathematics course (both are typically taught in year one of Computer Science university curricula or in the NOI/IOI training camps), and understand simple algorithmic analysis (at least the big-O notation). In the next subsections, we will address the different potential readers of this book.

To (Aspiring) IOI Contestants

IOI is not a speed contest and *for now*, currently excludes the topics listed in the following Table 1 (many are in Book 2). You can skip these topics until your University years (when you join that university's ICPC teams). However, learning these techniques in advance is definitely beneficial as some tasks in IOI can become easier with additional knowledge. Therefore, we recommend that you grab a copy of this book early in your competitive programming journey (i.e., during your high school days).

We are aware that one cannot win a medal in IOI just by mastering the contents of the *current version* (CP4) of this book. While we believe that many parts of the latest IOI syllabus [16] has been included in this book (especially Book 1)—hopefully enabling you to achieve a respectable score in future IOIs—we are well aware that modern IOI tasks require keen problem solving skills and tremendous creativity [20]—virtues that we cannot possibly impart through a static textbook. This book can provide knowledge, but the hard work must ultimately be done by you. With practice comes experience, and with experience comes skill. So, keep on practicing!

Topics in Book 2
Math: Big Integer, Modular Inverse, Probability Theory, Game Theory
String Processing: Suffix Trees/Arrays, KMP, String Hashing/Rabin-Karp
(Computational) Geometry: Various Geometry-specific library routines
Graph: Network Flow, Harder Matching problems, Rare NP-hard/complete Problems
More than half of the Rare Topics

Table 1: Not in IOI Syllabus [16] *Yet*

To Students of *Data Structures and lgorithms* Courses

The contents of this book have been expanded in CP4 so that the *first four* chapters of this book are more accessible to *first year* Computer Science students. Topics and exercises that we have found to be relatively difficult and thus unnecessarily discouraging for first timers have been moved to Book 2. This way, students who are new to Computer Science will perhaps not feel overly intimidated when they peruse Book 1.

Chapter 1 has a collection of very easy programming contest problems that can be solved by typical Computer Science students who have just passed (or currently taking) a basic programming methodology course.

Chapter 2 has received another major update. Now the writeups in the Sections about Linear and Non-linear Data Structures have been expanded with lots of written exercises so that this book can also be used to support a *Data Structures* course, especially in the terms of *implementation* details.

The four problem solving paradigms discussed in Chapter 3 appear frequently in typical *Algorithms* courses. The text in this chapter has been expanded and edited to help new Computer Science students.

Parts of Chapter 4 can also be used as a supplementary reading or *implementation* guide to enhance a *Discrete Mathematics* [50, 11] or a basic/intermediate (Graph) *Algorithms* course. We have also provide some (relatively) new insights on viewing Dynamic Programming techniques as algorithms on DAGs. Such discussion is currently still regrettably uncommon in many Computer Science textbooks.

To Job Seekers Preparing for IT Job Interview

It is well known that many job interviews in top IT companies involve fundamental data structure/algorithm/implementation questions. Many such questions have been discussed especially in Book 1 of CP4. We wish you the best in passing those interview(s).

On the other side of the job interview process, some interviewers read this book too in order to get inspiration for their interview questions.

To ICPC Contestants

You are the primary readers of this CP4. **Both** Book 1 and Book 2 are for you.

We know that one cannot probably win an ICPC Regional Contest just by mastering the contents of the *current version* of this book (CP4). While we have included a lot of materials in this book—much more than in the first three editions (CP1 $\subseteq$ CP2, then CP2 $\subseteq$ CP3, and finally CP3 $\subseteq$ CP4)—we are aware that much more than what this book can offer is required to achieve that feat. Some additional pointers to useful references are listed in the chapter notes for readers who are hungry for more. We believe, however, that your team will fare much better in future ICPCs after mastering the contents of this book. We hope that this book will serve as both inspiration and motivation for your 3-4 year journey competing in ICPCs during your University days.

To Teachers and Coaches

Wk	Topic	In CP4
01	Introduction	Chapter 1
02	Data Structures & Libraries	Chapter 2+9
03	Complete Search	Chapter 3+8+9
04	Dynamic Programming	Chapter 3+8+9
05	Buffer slot	Chapter 3/4/9/others
06	Mid-Semester Team Contest	Entire Book 1
-	Mid-Semester Break	-
07	Graph 1 (Network Flow)	Chapter 8+9
08	Graph 2 (Matching)	Chapter 8+9
09	NP-hard/complete Problems	Chapter 8
10	Mathematics	Chapter 5+9
11	String Processing (esp Suffix Array)	Chapter 6
12	(Computational) Geometry (Libraries)	Chapter 7+9
13	Final Team Contest	Entire Book 1+2 and beyond
-	No Final Examination	-

Table 2: Lesson Plan of Steven's CS3233 (ICPC Regionals Level)

This book is mainly used in Steven's CS3233 - "Competitive Programming" course in the School of Computing at the National University of Singapore. CS3233 is conducted in 13 teaching weeks using the lesson plan mentioned in Table 2 (we abbreviate "Week" as "Wk" in Table 2). Fellow teachers/coaches should feel free to modify the lesson plan to suit your students' needs. Hints or brief solutions of the **non-starred** written exercises in this book are given at the back of each chapter. Some of the **starred** written exercises are quite challenging and have neither hints nor solutions. These can probably be used as exam questions or for your local contest problems (of course, you have to solve them first!).

To All Readers

Due to its diversity of coverage and depth of discussion, this book is *not* meant to be read once, but several times. There are many written (≈ 258) and programming exercises (≈ 3458) listed and spread across almost every section. You can skip these exercises at first if the solution is too difficult or requires further knowledge and technique, and revisit them after studying other chapters of this book. Solving these exercises will strengthen your understanding of the concepts taught in this book as they usually involve interesting applications, twists or variants of the topic being discussed. Make an effort to attempt them—time spent solving these problems will definitely not be wasted.

We believe that this book is and will be relevant to many high school students, University students, and even for those who have graduated from University but still love problem solving using computers. Programming competitions such as the IOI and ICPC are here to stay, at least for many years ahead. New students should aim to understand and internalize the basic knowledge presented in this book before hunting for further challenges. However, the term 'basic' might be slightly misleading—please check the table of contents to understand what we mean by 'basic'.

As the title of this book may imply, the purpose of this book is clear: we aim to improve the reader's programming and problem solving abilities and thus increase the *lower bound* of programming competitions like the IOI and ICPC in the future. With more contestants mastering the contents of this book, we believe that the year 2010 (CP1 publication year) was a watershed marking an accelerated improvement in the standards of programming contests. We hope to help more contestants to achieve greater scores (≥ 70 – at least $\approx 6 \times 10$ points for solving all subtask 1 of the 6 tasks of the IOI) in future IOIs and help more teams solve more problems (≥ 2 – at least 1 more than the typical 1 giveaway problem per ICPC problemset) in future ICPCs. We also hope to see many IOI/ICPC coaches around the world adopt this book for the aid it provides in mastering topics that students cannot do without in competitive programming contests. If such a proliferation of the required 'lower-bound' knowledge for competitive programming is continued in this 2020s decade, then this book's primary objective of advancing the level of human knowledge will have been fulfilled, and we, as the authors of this book, will be very happy indeed.

Convention

There are lots of C/C++, Java, Python, and occasionally OCaml code included in this book. If they appear, they will be typeset in `this monospace font`. All code have 2 spaces per indentation level except Python code (4 spaces per indentation level).

For the C/C++ code in this book, we have adopted the frequent use of `typedef`s and macros—features that are commonly used by competitive programmers *for convenience, brevity, and coding speed*. However, we may not always be able to use those techniques in Java, Python, and/or OCaml as they may not contain similar or analogous features. Here are some examples of our C/C++ code shortcuts:

```
typedef long long ll;                          // common data types
typedef pair<int, int> ii;                     // comments that are mixed
typedef vector<int> vi;                        // in with code are placed
typedef vector<ii> vii;                        // on the right side
memset(memo, -1, sizeof memo);                 // to init DP memo table
vi memo(n, -1);                                // alternative way
memset(arr, 0, sizeof arr);                    // to clear array of ints
```

The following shortcuts are frequently used in both our C/C++ and Java code (not all of them are applicable in Python or OCaml):

```
// Shortcuts for "common" constants
const int INF = 1e9;                                  // 10^9 = 1B is < 2^31-1
const int LLINF = 4e18;                               // 4*10^18 is < 2^63-1
const double EPS = 1e-9;                              // a very small number
++i;                                                  // to simplify: i = i+1;
ans = a ? b : c;                                      // ternary operator
ans += val;                                           // from ans = ans+val;
index = (index+1) % n;                                // to right or back to 0
index = (index+n-1) % n;                              // to left or back to n-1
int ans = (int)((double)d + 0.5);                     // for rounding
ans = min(ans, new_computation);                      // min/max shortcut
// some code use short circuit && (AND) and || (OR)
// some code use structured bindings of C++17 for dealing with pairs/tuples
// we don't use braces for 1 liner selection/repetition body
// we use pass by reference (&) as far as possible
```

Problem Categorization

As of 19 July 2020, Steven, Felix, Suhendry—combined—have solved 2278 UVa problems ($\approx$ 45.88% of the entire UVa problemset as of publication date). Steven has also solved 5742.7 Kattis points ($\approx 1.4K$ other problems and $\approx$ 55.46% of the entire Kattis problemset as of publication date). There are $\approx$ 3458 problems have been categorized in this book.

These problems are categorized according to a "*load balancing*" scheme: if a problem can be classified into two or more categories, it will be placed in the category with a lower number of problems. This way, you may find that some problems have been 'wrongly' categorized, where the category that it appears in might not match the technique that you have used to solve it. We can only guarantee that if you see problem X in category Y, then you know that *we* have managed to solve problem X with the technique mentioned in the section that discusses category Y.

We have also limit each category to at most 35 (THIRTY-FIVE) problems, splitting them into separate categories when needed. In reality, each category has $\approx$ 17 problems on average. Thus, we have $\approx 3458/17 \approx 200+$ categories scattered throughout the book.

Utilize this categorization feature for your training! Solving at least a few problems from each category is a great way to diversify your problem solving skillset. For conciseness, we have limited ourselves to a maximum of 4 UVa + 3 Kattis (or 3 UVa + 4 Kattis) = 7 starred * (must try) problems per category and put the rest as extras (the hints for those extras can be read online at 'Methods to Solve' page of `https://cpbook.net`). You can say that you have 'somewhat mastered' CP4 if you have solved **at least three (3)** problems **per category** (this will take some time).

If you need hints for any of the problems (that we have solved), flip to the handy index at the back of this book instead of flipping through each chapter—it might save you some time. The index contains a list of UVa/Kattis problems, ordered by problem number/id (do a binary search!) and augmented by the pages that contain discussion of said problems (and the data structures and/or algorithms required to solve that problem). In CP4, we allow the hints to span more than one line (but not more than two lines) so that they can be a bit more meaningful. Of course you can always challenge yourself by *not* reading the hints first.

Changes for CP4

Competitive Programming textbook has been around since 2010 (first edition, dubbed as CP1), 2011 (second edition/CP2), and especially 2013 (third edition/CP3). There has been 7 years gap[2] between the release of CP3 to the release of this CP4 (just before the landmark IOI 2020 (Online) + IOI 2021 in Singapore). We highlight the important changes of these 7 years worth of additional Competitive Programming knowledge:

- Obviously, we have fixed all known typos, grammatical errors, and bugs that were found and reported by CP3 readers since 2013. It does not mean that this edition is 100% free from any bug though. We strive to have only very few errors in CP4.
- We have updated many sample code into C++17, Java 11, Python 3, and even some OCaml. Many of the sample code become simpler with a few more years of programming language update (e.g., C++17 structured binding declaration), the upgraded coding skills/styles of the authors, and various interesting contributions from our readers over these past few years.
- We use a public GitHub repo: `https://github.com/stevenhalim/cpbook-code` that contains the same sample code content as this book during the release date of this edition (19 July 2020). Obviously, the content of the GitHub repo will always be more up-to-date/complete than the printed version as time goes on. Please star, watch, fork, or even contribute to this public GitHub repo. You are free to use these source code for your next programming contest or any other purposes.
- We have added Kattis online judge `https://open.kattis.com` on top of UVa online judge and have raised the number of discussed problems to ≈ 3458. This is more than *two times* the number in CP3 (1675). Note that there are ≈ 150+ overlapping problems in both UVa and Kattis online judges. We only list them once (under Kattis problem id). Steven is top 9 (out of ≈ 141 132 users) in Kattis online judge and top 39 (out of ≈ 365 857) in UVa online judge as of 19 July 2020, i.e., at the 99.9th percentile for both online judges.
- We have digitized all hints of the ≈ 3458 problems that we have solved at `https://cpbook.net/methodstosolve`, including the extras that are not fully shown in the printed version of this book to save space. The online version has search/filter feature and will always be more up-to-date than the printed version as time goes on. The 750+ problems in Kattis online judge with the lowest points [1.1..3.5] as of 19 July 2020 have been solved by us and are discussed in this book.
- A few outdated problem categories have been adjusted/removed (e.g., Combining Max 1D/2D Range Sum, etc). A few/emerging problem categories have been opened (e.g., Pre-calculate-able, Try All Possible Answer(s), Fractions, NP-hard/complete, etc).
- To help our readers avoid the "needle in a haystack" issue, we *usually* select only top 4 UVa+3 Kattis (or top 3 UVa+4 Kattis), totalling 7 starred problems, per category. This reduce clutter and will help new competitive programmer to prioritize their training time on the better quality practice problems. This also saves a few precious pages that can be used to improve the actual content of the book.

[2]Including 10 ICPC Asia Regional Wins in between the release of CP3 (2013) and CP4 (2020).

- We have re-written almost every existing topic in the book to enhance their presentation. We have integrated our freely accessible `https://visualgo.net` algorithm visualization tool[3] as far as possible into this book. Obviously, the content shown in VisuAlgo will always be more up-to-date than the printed version as time goes on. All these new additions may be *subtle* but may be very important to avoid TLE/WA in the ever increasing difficulties of programming contest problems [17]. Many starred exercises in CP3 that are now deemed to be 'standard' by year 2020 have been integrated into the body text of this CP4 so do not be surprised to see a reduction of the number of written exercises in some chapters.

- Re-organization of topics compared to CP3, especially to facilitate the cleaner Book 1 versus Book 2 split:

  - Book 1 (Chapter 1-4)

    1. We select and organize some of the easiest problems found in UVa and Kattis online judges that were previously scattered in several chapters (especially from Chapter 5/6/7) into a compilation of exercises for those who have only started learning basic programming methodology in Chapter 1. It is now much easier to get the first few ACs in UVa and/or Kattis online judge(s) to kick start your Competitive Programming journey.
    2. We move basic string processing problems and some easier Ad Hoc string processing problems from Chapter 6 to Chapter 1 and highlight the usage of short Python code to deal with these problems.
    3. We move Roman numerals from Chapter 9 into Chapter 1, it is a rare but simple Ad Hoc problem.
    4. We move Inversion Index and Sorting in Linear Time from Chapter 9 into a 'Special Sorting Problems' sub-category in Chapter 2.
    5. We move Bracket Matching and Postfix Conversion/Calculator from Chapter 9 into a 'Special Stack-based Problems' sub-category in Chapter 2.
    6. We move basic Big Integer from Chapter 5 to Chapter 2 as it is essentially a built-in data structure for Python (3) and Java users (still classified as our-own library for C++ users). This way, readers can be presented with some easier Big Integer-related problems from the earlier chapters in Book 1.
    7. We move Order Statistics Tree from Chapter 9 as another non-linear data structure with its C++ specific `pbds` library in Chapter 2.
    8. We swap the order of two sections: Fenwick Tree (its basic form is much more easier to understand for beginners) and Segment Tree (more versatile).
    9. We move (Ad Hoc) Mathematics-related Complete Search problems from Chapter 5 to Chapter 3.
    10. We move Ad Hoc Josephus problem that mostly can be solved with Complete Search from Chapter 9 to Chapter 3.

  - With these content reorganizations, we are happy enough to declare that the content of Book 1 satisfy most[4] of the IOI syllabus [16] as of year 2020.

[3] VisuAlgo is built with modern web programming technologies, e.g., HTML5 SVG, canvas, CSS3, JavaScript (jQuery, D3.js library), PHP (Laravel framework), MySQL, etc. It has e-Lecture mode for basic explanations of various data structures and algorithms and Online Quiz mode to test basic understanding.

[4] Note that the IOI syllabus is an evolving document that is updated yearly.

- Book 2 (Chapter 5-9)

  1. We spread Java BigInteger specific features to related sections, e.g., Base number conversion and simplifying fractions with GCD at Ad Hoc mathematics section, probabilistic prime testing at Number Theory section, and modular exponentiation at Matrix Power section.
  2. We swap the order of two sections in Chapter 5: Number Theory (with the expanded modular arithmetic section) and Combinatorics (some harder Combinatorics problem now involve modular arithmetic).
  3. We swap the order of two sections in Chapter 6: String Processing with DP before String Matching. This is so that the discussion of String Matching spread across three related subsections: standard String Matching (KMP), Suffix Array, and String Matching with Hashing (Rabin-Karp).
  4. We reorganize the categorization of many DP problems that we have solved in Chapter 8.
  5. We defer the discussion of Network Flow from Chapter 4 (in CP3) to Chapter 8 (in CP4) as it is still excluded from the IOI Syllabus [16] as of year 2020.
  6. We move Graph Matching from Chapter 9 to Chapter 8, after the related Network Flow section and before the new section on NP-hard/complete problems.
  7. We add a new section on NP-complete decision and/or NP-hard optimization problems in Competitive Programming, compiling ideas that were previously scattered in CP3. We highlight that for such problem types, we are either given small instances (where Complete Search or Dynamic Programming is still sufficient) or the special case of the problem (where specialized polynomial/fast algorithm is still possible—including Greedy algorithm, Network Flow, or Graph Matching solutions).

- Chapter 1 changes:

  1. We add short writeups about the IOI and ICPC, the two important international programming competitions that use material in this book (and beyond).
  2. We include Python (3) as one of the supported programming languages in this book, especially for easier, non runtime-critical problems, Big Integer, and/or string processing problems. If you can save 5 minutes of coding time on your first Accepted solution and your team eventually solves 8 problems in the problem set, this is a saving of $8 \times 5 = 40$ total penalty minutes.
  3. We add *some* OCaml implementations (it is not yet used in the IOI or ICPC).
  4. We use up-to-date Competitive Programming techniques as of year 2020.

- Chapter 2 changes:

  1. Throughout this data structure chapter, we add much closer integration with our own freely accessible visualization tool: VisuAlgo.
  2. We add Python (3) and OCaml libraries on top of C++ STL and Java API.
  3. We significantly expand the discussion of Binary Heap, Hash Table, and (balanced) Binary Search Tree in Non-linear Data Structures section that are typically discussed in a "Data Structures and Algorithms" course.

4. We emphasize the usage of the faster Hash Tables (e.g., C++ `unordered_map`) instead of balanced BST (e.g., C++ `map`) if we do not need the ordering of keys and the keys are basic data types like integers or strings. We also recommend the simpler Direct Addressing Table (DAT) whenever it is applicable.
5. We highlight the usage of balanced BSTs as a powerful (but slightly slower) Priority Queue and as another sorting tool (Tree Sort).
6. We discuss ways to deal with graphs that are not labeled with [0..V-1] and on how to store some special graphs more efficiently.
7. We enhance the presentation of the UFDS data structure.
8. We add more features of Fenwick Tree data structure: Fenwick Tree as (a variant of) order statistics data structure, Range Update Point Query variant, and Range Update Range Query variant.
9. We add more feature of Segment Tree data structure: Range Update with Lazy Propagation to maintain its $O(\log n)$ performance.

- Chapter 3 changes:

1. We add two additional complete search techniques: Pre-calculate all (or some) answers and Try all possible answers (that cannot be binary-searched; or when the possible answers range is small). We also update iterative bitmask implementation to always use `LSOne` technique whenever possible. We also add more complete search tips, e.g., data compression to make the problem amenable to complete search techniques. We also tried Python for Complete Search problems. Although Python will mostly get TLE for harder Complete Search problems, there are ways to make Python usable for a few easier Complete Search problems.
2. We now favor implementation of Binary Search the Answer (BSTA) using for loop instead of while loop. We also integrate Ternary Search in this chapter.
3. We now consider greedy (bipartite) matching as another classic greedy problem. We add that some greedy algorithms use Priority Queue data structure to dynamically order the next candidates greedily.
4. We now use the $O(n \log k)$ LIS solution ('patience sort', not DP) as the default solution for modern LIS problem. We now use `LSOne` technique inside the $O(2^{n\ 1} \times n^2)$ DP-TSP solution to allow it to solve $n \leq$ [18..19] faster.

- Chapter 4 changes:

1. We redo almost all screenshots in this Chapter 4 using VisuAlgo tool.
2. We now set Kosaraju's algorithm as the default algorithm for finding Strongly Connected Components (SCCs) as it is simpler than Tarjan's algorithm.
3. We significantly expand the Shortest Paths section with many of its known variations. We discuss and compare both versions of Dijkstra's algorithm implementations. We move SPFA from Chapter 9, position this algorithm as Bellman-Ford 'extension', and called it as Bellman-Ford-Moore algorithm.
4. We significantly update the section on Euler graph and replace Fleury's algorithm with the better Hierholzer's algorithm.
5. We add remarks about a few more special (rare) graphs and their properties.

- Chapter 5 changes:

  1. We expand the discussion of this easy but big Ad Hoc Mathematics-related problems. We identify one more recurring Ad Hoc problems in Mathematics: Fraction.
  2. We recognize the shift of trend where the number of pure Big Integer problems is decreasing and the number of problems that require modular arithmetic is increasing. Therefore, the discussion on modular arithmetic in Number Theory section have been significantly expanded and presented earlier before being used in latter sections, e.g., Fermat's little theorem/modular multiplicative inverse is used in the implementation of Binomial Coefficients and Catalan Numbers in Combinatorics section, modular exponentiation is now the default in Matrix Power section.
  3. We expand the Combinatorics section with more review of counting techniques.
  4. We expand the discussion of Probability-related problems.
  5. We enhance the explanation of Floyd's (Tortoise-Hare) cycle-finding algorithm with VisuAlgo tool.
  6. We integrate Matrix Power into this chapter, expanded the writeup about matrix power, and integrate modular arithmetic techniques in this section.

- Chapter 6 changes:

  1. We discuss Digit DP as one more string processing problem with DP.
  2. We further strengthen our General Trie/Suffix Trie/Tree/Array explanation.
  3. We add String Hashing as alternative way to solve string processing related problems including revisiting the String Matching problem with hashing.
  4. We integrate and expand section on Anagram and Palindrome, both are classic string processing problems that have variants with varying difficulties.

- Chapter 7 changes:

  1. We further enhance the existing library routines, e.g., (the shorter to code) Andrew's Monotone Chain algorithm is now the default convex hull algorithm, replacing (the slightly longer to code and a bit slower) Graham's Scan.
  2. We redo the explanation and add VisuAlgo screenshots of algorithms on Polygon.

- Chapter 8 changes:

  1. We now set the faster $O(V^2 \times E)$ Dinic's algorithm as the default algorithm, replacing the slightly slower $O(V \times E^2)$ Edmonds-Karp algorithm. We also add a few more network flow applications.
  2. We expand the discussion of Graph Matching and its bipartite/non-bipartite + unweighted/weighted variants. We augment the Augmenting Path algorithm with the randomized greedy pre-processing step by default.
  3. We add a few more problem decomposition related techniques and listed many more such problems, ordered by their frequency of appearance.

- Chapter 9 changes:

  1. On top of enhancing previous writeups, we add more collection of new rare data structures, algorithms, and/or programming problems that have not been listed in the first eight chapters and *did not appear in CP3*. These new topics are:
     (a) Square Root Decomposition,
     (b) Heavy-Light Decomposition,
     (c) Tree Isomorphism,
     (d) De Bruijn Sequence,
     (e) Fast Fourier Transform,
     (f) Chinese Remainder Theorem,
     (g) Lucas' Theorem,
     (h) Combinatorial Game Theory,
     (i) Egg Dropping Puzzle,
     (j) Dynamic Programming Optimization,
     (k) Push-Relabel algorithm,
     (l) Kuhn-Munkres algorithm,
     (m) Edmonds' Matching algorithm,
     (n) Constructive Problem,
     (o) Interactive Problem,
     (p) Linear Programming,
     (q) Gradient Descent.

- In summary, someone who *only* master CP3 (published back in 2013) content can be easily beaten in a programming contest by someone who *only* master CP4 content (published in year 2020).

Supporting Websites

This book has an official companion web site at `https://cpbook.net`. The Methods to Solve tool is in this web site too.

We have also uploaded (almost) all source code discussed in this book in the public GitHub repository of this book: `https://github.com/stevenhalim/cpbook-code`.

Since the third edition of this book, many data structures and algorithms discussed in this book already have interactive visualizations at `https://visualgo.net`.

All UVa Online Judge programming exercises in this book have been integrated in the `https://uhunt.onlinejudge.org/` tool.

All Kattis Online Judge programming exercises in this book can be easily accessed using the "Kattis Hint Giver" Google Chrome extension (created by Steven's student Lin Si Jie) that integrates the content of Methods to Solve directly to Kattis's problems pages.

Copyright

To a better future of humankind,
Steven Halim, Felix Halim, and Suhendry Effendy
Singapore, 19 July 2020