## Contents (detailed)

Preface 19

0.1 To the student 21
  0.1.1 Is the effort worth it? 21
0.2 To potential instructors 22
0.3 Acknowledgements 24

Preliminaries 27

0 Introduction 29
  0.1 Integer multiplication: an example of an algorithm 30
  0.2 Extended Example: A faster way to multiply (optional) 33
  0.3 Algorithms beyond arithmetic 37
  0.4 On the importance of negative results. 38
  0.5 Roadmap to the rest of this book 39
    0.5.1 Dependencies between chapters 40
  0.6 Exercises 42
  0.7 Bibliographical notes 44

1 Mathematical Background 47
  1.1 This chapter: a reader’s manual 47
  1.2 A quick overview of mathematical prerequisites 48
  1.3 Reading mathematical texts 49
    1.3.1 Definitions 50
    1.3.2 Assertions: Theorems, lemmas, claims 50
    1.3.3 Proofs 50
  1.4 Basic discrete math objects 51
    1.4.1 Sets 51
    1.4.2 Special sets 52
    1.4.3 Functions 54
    1.4.4 Graphs 56
    1.4.5 Logic operators and quantifiers 59
    1.4.6 Quantifiers for summations and products 60
    1.4.7 Parsing formulas: bound and free variables 60
    1.4.8 Asymptotics and Big-O notation 62
1.4.9 Some “rules of thumb” for Big-O notation . . . 63
1.5 Proofs ........................................ 64
  1.5.1 Proofs and programs ...................... 65
  1.5.2 Proof writing style ....................... 65
  1.5.3 Patterns in proofs ....................... 66
1.6 Extended example: Topological Sorting .......... 69
  1.6.1 Mathematical induction .................. 70
  1.6.2 Proving the result by induction .......... 71
  1.6.3 Minimality and uniqueness ............... 73
1.7 This book: notation and conventions .......... 75
  1.7.1 Variable name conventions ............... 76
  1.7.2 Some idioms ............................. 77
1.8 Exercises .................................... 79
1.9 Bibliographical notes ........................ 81

2 Computation and Representation 83
  2.1 Defining representations ................... 84
    2.1.1 Representing natural numbers .......... 85
    2.1.2 Meaning of representations (discussion). . 87
  2.2 Representations beyond natural numbers .... 88
    2.2.1 Representing (potentially negative) integers .. 88
    2.2.2 Two’s complement representation (optional) . 89
    2.2.3 Rational numbers, and representing pairs of strings ................................. 90
  2.3 Representing real numbers ................... 91
    2.3.1 Can we represent reals exactly? .......... 92
  2.4 Representing objects beyond numbers ....... 98
    2.4.1 Finite representations .................. 99
    2.4.2 Prefix-free encoding .................... 99
    2.4.3 Making representations prefix-free .... 102
    2.4.4 “Proof by Python” (optional) ......... 103
    2.4.5 Representing letters and text .......... 105
    2.4.6 Representing vectors, matrices, images .. 107
    2.4.7 Representing graphs .................... 107
    2.4.8 Representing lists and nested lists .... 107
    2.4.9 Notation ................................ 107
  2.5 Defining computational tasks as mathematical functions 108
    2.5.1 Distinguish functions from programs! ... 110
  2.6 Exercises .................................... 112
  2.7 Bibliographical notes ........................ 116

I Finite computation 119

3 Defining computation 121
3.1  Defining computation ........................................ 122
3.2  Computing using AND, OR, and NOT ..................... 124
   3.2.1  Some properties of AND and OR .................... 126
   3.2.2  Extended example: Computing XOR from
          AND, OR, and NOT .................................. 126
   3.2.3  Informally defining “basic operations” and
          “algorithms” ........................................... 129
3.3  Boolean Circuits .............................................. 130
   3.3.1  Boolean circuits: a formal definition ............... 131
   3.3.2  Equivalence of circuits and straight-line programs 134
3.4  Physical implementations of computing devices (di-
     gression) .................................................... 138
   3.4.1  Transistors ............................................ 138
   3.4.2  Logical gates from transistors ...................... 139
   3.4.3  Biological computing ................................. 139
   3.4.4  Cellular automata and the game of life ............ 139
   3.4.5  Neural networks ..................................... 140
   3.4.6  A computer made from marbles and pipes .......... 140
3.5  The NAND function ........................................... 141
   3.5.1  NAND Circuits ........................................ 142
   3.5.2  More examples of NAND circuits (optional) ....... 144
   3.5.3  The NAND-CIRC Programming language ............ 145
3.6  Equivalence of all these models ............................ 147
   3.6.1  Circuits with other gate sets ....................... 148
   3.6.2  Specification vs. implementation (again) ........ 149
3.7  Exercises ..................................................... 151
3.8  Biographical notes .......................................... 153

4  Syntactic sugar, and computing every function ........ 155
   4.1  Some examples of syntactic sugar ....................... 156
      4.1.1  User-defined procedures .......................... 156
      4.1.2  Proof by Python (optional) ....................... 159
      4.1.3  Conditional statements ............................ 161
   4.2  Extended example: Addition and Multiplication (op-
        tional) ................................................. 162
   4.3  The LOOKUP function .................................... 164
      4.3.1  Constructing a NAND-CIRC program for
             LOOKUP ........................................... 164
   4.4  Computing every function ................................ 166
      4.4.1  Proof of NAND’s Universality ..................... 167
      4.4.2  Improving by a factor of $n$ (optional) ......... 169
   4.5  Computing every function: An alternative proof ...... 170
   4.6  The class $SIZE(T)$ ..................................... 172
   4.7  Exercises ................................................ 175
4.8 Bibliographical notes ........................................ 179

5 Code as data, data as code .................................. 181
  5.1 Representing programs as strings ....................... 182
  5.2 Counting programs, and lower bounds on the size of NAND-CIRC programs ....................... 184
    5.2.1 Size hierarchy theorem (optional) ............... 186
  5.3 The tuples representation ............................... 188
    5.3.1 From tuples to strings ............................ 189
  5.4 A NAND-CIRC interpreter in NAND-CIRC .......... 190
    5.4.1 Efficient universal programs ................. 191
    5.4.2 A NAND-CIRC interpreter in “pseudocode” .... 192
    5.4.3 A NAND interpreter in Python .............. 194
    5.4.4 Constructing the NAND-CIRC interpreter in NAND-CIRC .................. 194
  5.5 A Python interpreter in NAND-CIRC (discussion) .... 197
  5.6 The physical extended Church-Turing thesis (discussion) ........................................ 199
    5.6.1 Attempts at refuting the PECTT ............... 201
  5.7 Recap of Part I: Finite Computation ............... 206
  5.8 Exercises ............................................. 207
  5.9 Bibliographical notes ................................ 209

II Uniform computation .................................. 211

6 Loops and infinity .................................... 213
  6.1 Turing Machines ...................................... 216
    6.1.1 Extended example: A Turing machine for palindromes ........................................ 217
    6.1.2 Turing machines: a formal definition ........ 219
    6.1.3 Computable functions ......................... 220
    6.1.4 Infinite loops and partial functions ........ 222
  6.2 Turing machines as programming languages .......... 223
    6.2.1 The NAND-TM Programming language .......... 224
    6.2.2 Sneak peak: NAND-TM vs Turing machines ... 227
    6.2.3 Examples ........................................ 228
  6.3 Equivalence of Turing machines and NAND-TM programs .................................................. 230
    6.3.1 Specification vs implementation (again) .... 233
  6.4 NAND-TM syntactic sugar ............................ 234
    6.4.1 “GOTO” and inner loops ...................... 235
  6.5 Uniformity, and NAND vs NAND-TM (discussion) .... 237
  6.6 Exercises ............................................. 238
  6.7 Bibliographical notes ................................ 241
### 7 Equivalent models of computation

#### 7.1 RAM machines and NAND-RAM

- Indexed access in NAND-TM
- Two dimensional arrays in NAND-TM
- All the rest

#### 7.2 The gory details (optional)

- Indexed access in NAND-TM
- Two dimensional arrays in NAND-TM
- All the rest

#### 7.3 Turing equivalence (discussion)

- The “Best of both worlds” paradigm
- Let’s talk about abstractions
- Turing completeness and equivalence, a formal definition (optional)

#### 7.4 Cellular automata

- One dimensional cellular automata are Turing complete
- Configurations of Turing machines and the next-step function

#### 7.5 Lambda calculus and functional programming languages

- Applying functions to functions
- Obtaining multi-argument functions via Currying
- Formal description of the λ calculus
- Infinite loops in the λ calculus

#### 7.6 The “Enhanced” λ calculus

- Computing a function in the enhanced λ calculus
- Enhanced λ calculus is Turing-complete

#### 7.7 From enhanced to pure λ calculus

- List processing
- The Y combinator, or recursion without recursion

#### 7.8 The Church-Turing Thesis (discussion)

- Different models of computation

#### 7.9 Exercises

#### 7.10 Bibliographical notes

### 8 Universality and uncomputability

#### 8.1 Universality or a meta-circular evaluator

- Proving the existence of a universal Turing Machine
- Implications of universality (discussion)

#### 8.2 Is every function computable?

#### 8.3 The Halting problem

- Is the Halting problem really hard? (discussion)
- A direct proof of the uncomputability of HALT (optional)

#### 8.4 Reductions
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.6.2</td>
<td>“Unrolling the loop”: algorithmic transformation of Turing Machines to circuits</td>
<td>423</td>
</tr>
<tr>
<td>12.6.3</td>
<td>Can uniform algorithms simulate non uniform ones?</td>
<td>426</td>
</tr>
<tr>
<td>12.6.4</td>
<td>Uniform vs. Nonuniform computation: A recap</td>
<td>427</td>
</tr>
<tr>
<td>12.7</td>
<td>Exercises</td>
<td>428</td>
</tr>
<tr>
<td>12.8</td>
<td>Bibliographical notes</td>
<td>431</td>
</tr>
<tr>
<td>13</td>
<td>Polynomial-time reductions</td>
<td>433</td>
</tr>
<tr>
<td>13.1</td>
<td>Formal definitions of problems</td>
<td>434</td>
</tr>
<tr>
<td>13.2</td>
<td>Polynomial-time reductions</td>
<td>435</td>
</tr>
<tr>
<td>13.3</td>
<td>Reducing 3SAT to zero one equations</td>
<td>437</td>
</tr>
<tr>
<td>13.3.1</td>
<td>Quadratic equations</td>
<td>441</td>
</tr>
<tr>
<td>13.4</td>
<td>The independent set problem</td>
<td>442</td>
</tr>
<tr>
<td>13.5</td>
<td>Reducing Independent Set to Maximum Cut</td>
<td>445</td>
</tr>
<tr>
<td>13.6</td>
<td>Reducing 3SAT to Longest Path</td>
<td>447</td>
</tr>
<tr>
<td>13.6.1</td>
<td>Summary of relations</td>
<td>449</td>
</tr>
<tr>
<td>13.7</td>
<td>Exercises</td>
<td>450</td>
</tr>
<tr>
<td>13.8</td>
<td>Bibliographical notes</td>
<td>450</td>
</tr>
<tr>
<td>14</td>
<td>NP, NP completeness, and the Cook-Levin Theorem</td>
<td>451</td>
</tr>
<tr>
<td>14.1</td>
<td>The class NP</td>
<td>451</td>
</tr>
<tr>
<td>14.1.1</td>
<td>Examples of functions in NP</td>
<td>454</td>
</tr>
<tr>
<td>14.1.2</td>
<td>Basic facts about NP</td>
<td>456</td>
</tr>
<tr>
<td>14.2</td>
<td>From NP to 3SAT: The Cook-Levin Theorem</td>
<td>458</td>
</tr>
<tr>
<td>14.2.1</td>
<td>What does this mean?</td>
<td>459</td>
</tr>
<tr>
<td>14.2.2</td>
<td>The Cook-Levin Theorem: Proof outline</td>
<td>460</td>
</tr>
<tr>
<td>14.3</td>
<td>The NANDSAT Problem, and why it is NP hard</td>
<td>461</td>
</tr>
<tr>
<td>14.4</td>
<td>The 3NAND problem</td>
<td>462</td>
</tr>
<tr>
<td>14.5</td>
<td>From 3NAND to 3SAT</td>
<td>465</td>
</tr>
<tr>
<td>14.6</td>
<td>Wrapping up</td>
<td>466</td>
</tr>
<tr>
<td>14.7</td>
<td>Exercises</td>
<td>468</td>
</tr>
<tr>
<td>14.8</td>
<td>Bibliographical notes</td>
<td>469</td>
</tr>
<tr>
<td>15</td>
<td>What if P equals NP?</td>
<td>471</td>
</tr>
<tr>
<td>15.1</td>
<td>Search-to-decision reduction</td>
<td>472</td>
</tr>
<tr>
<td>15.2</td>
<td>Optimization</td>
<td>474</td>
</tr>
<tr>
<td>15.2.1</td>
<td>Example: Supervised learning</td>
<td>478</td>
</tr>
<tr>
<td>15.2.2</td>
<td>Example: Breaking cryptosystems</td>
<td>479</td>
</tr>
<tr>
<td>15.3</td>
<td>Finding mathematical proofs</td>
<td>479</td>
</tr>
<tr>
<td>15.4</td>
<td>Quantifier elimination (advanced)</td>
<td>480</td>
</tr>
<tr>
<td>15.4.1</td>
<td>Application: self improving algorithm for 3SAT</td>
<td>483</td>
</tr>
<tr>
<td>15.5</td>
<td>Approximating counting problems and posterior sampling (advanced, optional)</td>
<td>483</td>
</tr>
<tr>
<td>15.6</td>
<td>What does all of this imply?</td>
<td>484</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>19.0.2 Amplification</td>
<td>526</td>
<td></td>
</tr>
<tr>
<td>19.1 BPP and NP completeness</td>
<td>527</td>
<td></td>
</tr>
<tr>
<td>19.2 The power of randomization</td>
<td>529</td>
<td></td>
</tr>
<tr>
<td>19.2.1 Solving BPP in exponential time</td>
<td>529</td>
<td></td>
</tr>
<tr>
<td>19.2.2 Simulating randomized algorithms by circuits or straight-line programs</td>
<td>529</td>
<td></td>
</tr>
<tr>
<td>19.3 Derandomization</td>
<td>531</td>
<td></td>
</tr>
<tr>
<td>19.3.1 Pseudorandom generators</td>
<td>532</td>
<td></td>
</tr>
<tr>
<td>19.3.2 From existence to constructivity</td>
<td>534</td>
<td></td>
</tr>
<tr>
<td>19.3.3 Usefulness of pseudorandom generators</td>
<td>535</td>
<td></td>
</tr>
<tr>
<td>19.4 $P = NP$ and BPP vs $P$</td>
<td>536</td>
<td></td>
</tr>
<tr>
<td>19.5 Non-constructive existence of pseudorandom generators (advanced, optional)</td>
<td>539</td>
<td></td>
</tr>
<tr>
<td>19.6 Exercises</td>
<td>542</td>
<td></td>
</tr>
<tr>
<td>19.7 Bibliographical notes</td>
<td>542</td>
<td></td>
</tr>
<tr>
<td>19.8 Further explorations</td>
<td>542</td>
<td></td>
</tr>
<tr>
<td>19.9 Acknowledgements</td>
<td>542</td>
<td></td>
</tr>
<tr>
<td>V Advanced topics</td>
<td>543</td>
<td></td>
</tr>
<tr>
<td>20 Cryptography</td>
<td>545</td>
<td></td>
</tr>
<tr>
<td>20.1 Classical cryptosystems</td>
<td>546</td>
<td></td>
</tr>
<tr>
<td>20.2 Defining encryption</td>
<td>549</td>
<td></td>
</tr>
<tr>
<td>20.3 Defining security of encryption</td>
<td>549</td>
<td></td>
</tr>
<tr>
<td>20.4 Perfect secrecy</td>
<td>551</td>
<td></td>
</tr>
<tr>
<td>20.4.1 Example: Perfect secrecy in the battlefield</td>
<td>552</td>
<td></td>
</tr>
<tr>
<td>20.4.2 Constructing perfectly secret encryption</td>
<td>553</td>
<td></td>
</tr>
<tr>
<td>20.5 Necessity of long keys</td>
<td>554</td>
<td></td>
</tr>
<tr>
<td>20.6 Computational secrecy</td>
<td>556</td>
<td></td>
</tr>
<tr>
<td>20.6.1 Stream ciphers or the “derandomized one-time pad”</td>
<td>558</td>
<td></td>
</tr>
<tr>
<td>20.7 Computational secrecy and NP</td>
<td>560</td>
<td></td>
</tr>
<tr>
<td>20.8 Public key cryptography</td>
<td>562</td>
<td></td>
</tr>
<tr>
<td>20.8.1 Defining public key encryption</td>
<td>564</td>
<td></td>
</tr>
<tr>
<td>20.8.2 Diffie-Hellman key exchange</td>
<td>565</td>
<td></td>
</tr>
<tr>
<td>20.9 Other security notions</td>
<td>567</td>
<td></td>
</tr>
<tr>
<td>20.10 Magic</td>
<td>567</td>
<td></td>
</tr>
<tr>
<td>20.10.1 Zero knowledge proofs</td>
<td>567</td>
<td></td>
</tr>
<tr>
<td>20.10.2 Fully homomorphic encryption</td>
<td>568</td>
<td></td>
</tr>
<tr>
<td>20.10.3 Multiparty secure computation</td>
<td>568</td>
<td></td>
</tr>
<tr>
<td>20.11 Exercises</td>
<td>570</td>
<td></td>
</tr>
<tr>
<td>20.12 Bibliographical notes</td>
<td>570</td>
<td></td>
</tr>
<tr>
<td>20.13 Further explorations</td>
<td>570</td>
<td></td>
</tr>
</tbody>
</table>