



HAL
open science

A Guide to Algorithm Design: Paradigms, Methods, and Complexity Analysis

Anne Benoit, Yves Robert, Frédéric Vivien

► **To cite this version:**

Anne Benoit, Yves Robert, Frédéric Vivien. A Guide to Algorithm Design: Paradigms, Methods, and Complexity Analysis. Chapman & Hall/CRC, pp.380, 2013, Applied Algorithms and Data Structures series, 9781439825648. hal-00908448

HAL Id: hal-00908448

<https://inria.hal.science/hal-00908448>

Submitted on 23 Nov 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Anne Benoit, Yves Robert, Frédéric Vivien

A guide to algorithm design

Paradigms, methods and complexity analysis

CRC PRESS

Boca Raton London New York Washington, D.C.

Contents

Preface	xv
I Polynomial-time algorithms: Exercises	1
1 Introduction to complexity	3
1.1 On the complexity to compute x^n	3
1.1.1 Naive method	4
1.1.2 Binary method	4
1.1.3 Factorization method	4
1.1.4 Knuth's tree method	5
1.1.5 Complexity results	6
1.2 Asymptotic notations: O , o , Θ , and Ω	8
1.3 Exercises	8
Exercise 1.1: Longest balanced section	8
Exercise 1.2: Find the star	9
Exercise 1.3: Breaking boxes	9
Exercise 1.4: Maximum of n integers	10
Exercise 1.5: Maximum and minimum of n integers	10
Exercise 1.6: Maximum and second maximum of n integers	11
Exercise 1.7: Merging two sorted sets	11
Exercise 1.8: The toolbox	12
Exercise 1.9: Sorting a small number of objects	12
1.4 Solutions to exercises	14
Solution to Exercise 1.1: Longest balanced section	14
Solution to Exercise 1.2: Find the star	15
Solution to Exercise 1.3: Breaking boxes	16
Solution to Exercise 1.4: Maximum of n integers	17
Solution to Exercise 1.5: Maximum and minimum of n integers	20
Solution to Exercise 1.6: Maximum and second maximum of n integers	23
Solution to Exercise 1.7: Merging two sorted sets	25
Solution to Exercise 1.8: The toolbox	26
Solution to Exercise 1.9: Sorting a small number of objects	29
1.5 Bibliographical notes	31

2	Divide-and-conquer	33
2.1	Strassen's algorithm	33
2.2	Master theorem	36
2.3	Solving recurrences	37
2.3.1	Solving homogeneous recurrences	37
2.3.2	Solving nonhomogeneous recurrences	38
2.3.3	Solving the recurrence for Strassen's algorithm	39
2.4	Exercises	39
	Exercise 2.1: Product of two polynomials	39
	Exercise 2.2: Toeplitz matrices	40
	Exercise 2.3: Maximum sum	40
	Exercise 2.4: Boolean matrices: The Four-Russians algorithm	41
	Exercise 2.5: Matrix multiplication and inversion	42
2.5	Solutions to exercises	42
	Solution to Exercise 2.1: Product of two polynomials	42
	Solution to Exercise 2.2: Toeplitz matrices	44
	Solution to Exercise 2.3: Maximum sum	45
	Solution to Exercise 2.4: Boolean matrices: The Four-Russians algorithm	49
	Solution to Exercise 2.5: Matrix multiplication and inversion	50
2.6	Bibliographical notes	51
3	Greedy algorithms	53
3.1	Motivating example: The sports hall	53
3.2	Designing greedy algorithms	55
3.3	Graph coloring	56
3.3.1	On coloring bipartite graphs	56
3.3.2	Greedy algorithms to color general graphs	57
3.3.3	Coloring interval graphs	60
3.4	Theory of matroids	61
3.5	Exercises	64
	Exercise 3.1: Interval cover	64
	Exercise 3.2: Memory usage	64
	Exercise 3.3: Scheduling dependent tasks on several machines	65
	Exercise 3.4: Scheduling independent tasks with priorities	66
	Exercise 3.5: Scheduling independent tasks with deadlines	66
	Exercise 3.6: Edge matroids	67
	Exercise 3.7: Huffman code	67
3.6	Solutions to exercises	68
	Solution to Exercise 3.1: Interval cover	68
	Solution to Exercise 3.2: Memory usage	69
	Solution to Exercise 3.3: Scheduling dependent tasks on several machines	71
	Solution to Exercise 3.4: Scheduling independent tasks with priorities	72

Solution to Exercise 3.5: Scheduling independent tasks with deadlines	73
Solution to Exercise 3.6: Edge matroids	74
Solution to Exercise 3.7: Huffman code	75
3.7 Bibliographical notes	79
4 Dynamic programming	81
4.1 The coin changing problem	81
4.2 The knapsack problem	84
4.3 Designing dynamic-programming algorithms	86
4.4 Exercises	87
Exercise 4.1: Matrix chains	87
Exercise 4.2: The library	88
Exercise 4.3: Polygon triangulation	88
Exercise 4.4: Square of ones	89
Exercise 4.5: The wind band	89
Exercise 4.6: Ski rental	89
Exercise 4.7: Building set	90
4.5 Solutions to exercises	90
Solution to Exercise 4.1: Matrix chains	90
Solution to Exercise 4.2: The library	91
Solution to Exercise 4.3: Polygon triangulation	93
Solution to Exercise 4.4: Square of ones	96
Solution to Exercise 4.5: The wind band	98
Solution to Exercise 4.6: Ski rental	98
Solution to Exercise 4.7: Building set	102
4.6 Bibliographical notes	103
5 Amortized analysis	105
5.1 Methods for amortized analysis	105
5.1.1 Running examples	105
5.1.2 Aggregate analysis	106
5.1.3 Accounting method	106
5.1.4 Potential method	107
5.2 Exercises	108
Exercise 5.1: Binary counter	108
Exercise 5.2: Inserting and deleting	108
Exercise 5.3: Stack	109
Exercise 5.4: Deleting half the elements	109
Exercise 5.5: Searching and inserting	109
Exercise 5.6: Splay trees	110
Exercise 5.7: Half perimeter of a polygon	112
5.3 Solutions to exercises	112
Solution to Exercise 5.1: Binary counter	112
Solution to Exercise 5.2: Inserting and deleting	113

Solution to Exercise 5.3: Stack	114
Solution to Exercise 5.4: Deleting half the elements	115
Solution to Exercise 5.5: Searching and inserting	116
Solution to Exercise 5.6: Splay trees	117
Solution to Exercise 5.7: Half perimeter of a polygon	119
5.4 Bibliographical notes	122
II NP-completeness and beyond	123
6 NP-completeness	125
6.1 A practical approach to complexity theory	125
6.2 Problem classes	126
6.2.1 Problems in P	127
6.2.2 Problems in NP	129
6.3 NP-complete problems and reduction theory	132
6.3.1 Polynomial reduction	132
6.3.2 Cook's theorem	133
6.3.3 Growing the class NPC of NP-complete problems	134
6.3.4 Optimization problems versus decision problems	135
6.4 Examples of NP-complete problems and reductions	136
6.4.1 3-SAT	136
6.4.2 CLIQUE	138
6.4.3 VERTEX-COVER	139
6.4.4 Scheduling problems	140
6.4.5 Other famous NP-complete problems	142
6.5 Importance of problem definition	143
6.6 Strong NP-completeness	145
6.7 Why does it matter?	146
6.8 Bibliographical notes	146
7 Exercises on NP-completeness	149
7.1 Easy reductions	149
Exercise 7.1: Wheel	149
Exercise 7.2: Knights of the round table	149
Exercise 7.3: Variants of CLIQUE	149
Exercise 7.4: Path with vertex pairs	150
Exercise 7.5: VERTEX-COVER with even degrees	150
Exercise 7.6: Around 2-PARTITION	150
7.2 About graph coloring	151
Exercise 7.7: COLOR	151
Exercise 7.8: 3-COLOR	151
Exercise 7.9: 3-COLOR-PLAN	152
7.3 Scheduling problems	152
Exercise 7.10: Scheduling independent tasks with p processors	152

	Exercise 7.11: Scheduling with two processors	152
7.4	More involved reductions	153
	Exercise 7.12: Transitive subchain	153
	Exercise 7.13: INDEPENDENT SET	153
	Exercise 7.14: DOMINATING SET	153
	Exercise 7.15: Carpenter	153
	Exercise 7.16: k -center	153
	Exercise 7.17: Variants of 3-SAT	154
	Exercise 7.18: Variants of SAT	154
7.5	2-PARTITION is NP-complete	155
	Exercise 7.19: SUBSET-SUM	155
	Exercise 7.20: NP-completeness of 2-PARTITION	155
7.6	Solutions to exercises	155
	Solution to Exercise 7.1: Wheel	156
	Solution to Exercise 7.2: Knights of the round table	156
	Solution to Exercise 7.3: Variants of CLIQUE	157
	Solution to Exercise 7.4: Path with vertex pairs	158
	Solution to Exercise 7.5: VERTEX-COVER with even degrees	158
	Solution to Exercise 7.6: Around 2-PARTITION	159
	Solution to Exercise 7.7: COLOR	160
	Solution to Exercise 7.8: 3-COLOR	162
	Solution to Exercise 7.9: 3-COLOR-PLAN	163
	Solution to Exercise 7.10: Scheduling independent tasks with p processors	166
	Solution to Exercise 7.11: Scheduling with two processors	166
	Solution to Exercise 7.12: Transitive subchain	167
	Solution to Exercise 7.13: INDEPENDENT SET	168
	Solution to Exercise 7.14: DOMINATING SET	169
	Solution to Exercise 7.15: Carpenter	170
	Solution to Exercise 7.16: k -center	171
	Solution to Exercise 7.17: Variants of 3-SAT	172
	Solution to Exercise 7.18: Variants of SAT	174
	Solution to Exercise 7.19: SUBSET-SUM	175
	Solution to Exercise 7.20: NP-completeness of 2-PARTITION	177
7.7	Bibliographical notes	178
8	Beyond NP-completeness	179
8.1	Approximation results	179
	8.1.1 Approximation algorithms	180
	8.1.2 Vertex cover	181
	8.1.3 Traveling salesman problem (TSP)	182
	8.1.4 Bin packing	183
	8.1.5 2-PARTITION	187
8.2	Polynomial problem instances	192
	8.2.1 Partitioning problems	193

8.2.2	Assessing problem complexity	194
8.3	Linear programming	195
8.3.1	Formal definition	195
8.3.2	Relaxation and rounding	197
8.4	Randomized algorithms	200
8.4.1	The algorithm	201
8.4.2	Results	201
8.5	Branch-and-bound and backtracking	202
8.5.1	Backtracking: The n queens	203
8.5.2	Branch-and-bound: The knapsack	204
8.5.3	Graph algorithms	206
8.6	Bibliographical notes	209
9	Exercises going beyond NP-completeness	211
9.1	Approximation results	211
	Exercise 9.1: Single machine scheduling	211
	Exercise 9.2: SUBSET-SUM	212
	Exercise 9.3: SET-COVER	213
	Exercise 9.4: VERTEX-COVER	213
	Exercise 9.5: Scheduling independent tasks in parallel	215
	Exercise 9.6: Point clustering	215
	Exercise 9.7: k -center	216
	Exercise 9.8: Knapsack	217
9.2	Dealing with NP-complete problems	218
	Exercise 9.9: Mixed integer linear program for replica placement	218
	Exercise 9.10: A randomized algorithm for independent set	218
	Exercise 9.11: Branch-and-bound applied to MAX-SAT	219
9.3	Solutions to exercises	219
	Solution to Exercise 9.1: Single machine scheduling	219
	Solution to Exercise 9.2: SUBSET-SUM	221
	Solution to Exercise 9.3: SET-COVER	223
	Solution to Exercise 9.4: VERTEX-COVER	224
	Solution to Exercise 9.5: Scheduling independent tasks in parallel	226
	Solution to Exercise 9.6: Point clustering	228
	Solution to Exercise 9.7: k -center	229
	Solution to Exercise 9.8: Knapsack	231
	Solution to Exercise 9.9: Mixed integer linear program for replica placement	234
	Solution to Exercise 9.10: A randomized algorithm for independent set	237
	Solution to Exercise 9.11: Branch-and-bound applied to MAX-SAT	237
9.4	Bibliographical notes	238

III Reasoning on problem complexity	239
10 Reasoning to assess a problem complexity	241
10.1 Basic reasoning	241
10.1.1 Polynomial instances	241
10.1.2 NP-complete instances	242
10.2 Set of problems with polynomial-time algorithms	243
10.3 Set of NP-complete problems	244
10.3.1 Numbers	245
10.3.2 Graphs	246
11 Chains-on-chains partitioning	249
11.1 Optimal algorithms for homogeneous resources	249
11.1.1 Dynamic-programming algorithm	250
11.1.2 Binary search algorithm	250
11.1.3 Improved algorithms	250
11.2 Variants of the problem	252
11.2.1 Communication costs	252
11.2.2 Chain of heterogeneous resources	253
11.3 Extension to a clique of heterogeneous resources	254
11.3.1 NP-completeness	254
11.3.2 Practical solutions	257
11.3.3 Integer linear program	257
11.4 Conclusion	258
12 Replica placement in tree networks	261
12.1 Access policies	262
12.1.1 Motivation	262
12.1.2 Impact of the policies on the existence of a solution	263
12.1.3 Impact of the policies on the cost of a solution	264
12.2 Complexity results	266
12.2.1 Definitions	266
12.2.2 MINNB problem	267
12.2.3 MINCOST problem	273
12.2.4 Integer linear program	275
12.3 Variants of the replica placement problem	279
12.3.1 Enforcing a quality of service	280
12.3.2 Power-aware replica placement	282
12.4 Conclusion	286
13 Packet routing	287
13.1 MEDP: Maximum edge-disjoint paths	288
13.1.1 Problem statement	288
13.1.2 Naive greedy algorithm	289
13.1.3 Short-requests-first greedy algorithm	291

13.1.4	Inapproximability result	292
13.2	PRVP: Packet routing with variable-paths	294
13.2.1	Problem statement	294
13.2.2	Bounding optimal makespan via linear programming	295
13.2.3	Routing algorithm	297
13.2.4	Steady-state approach	300
13.3	Conclusion	301
14	Matrix product, or tiling the unit square	303
14.1	Problem motivation	304
14.2	NP-completeness	307
14.3	A guaranteed heuristic	311
14.3.1	The COLPERISUM(s) problem	312
14.3.2	Performance guarantee	316
14.3.3	Looking for a better solution	317
14.4	Related problems	320
15	Online scheduling	321
15.1	Flow time optimization	322
15.2	Competitive analysis	324
15.2.1	Definition	324
15.2.2	Method to establish a competitive analysis result	327
15.3	Makespan optimization	334
15.3.1	List scheduling algorithms	335
15.3.2	Randomized optimization of makespan	338
15.4	Conclusion	347
	References	349
	Index	359

List of exercises

1.1	Longest balanced section	8
1.2	Find the star	9
1.3	Breaking boxes	9
1.4	Maximum of n integers	10
1.5	Maximum and minimum of n integers	10
1.6	Maximum and second maximum of n integers	11
1.7	Merging two sorted sets	11
1.8	The toolbox	12
1.9	Sorting a small number of objects	12
2.1	Product of two polynomials	39
2.2	Toeplitz matrices	40
2.3	Maximum sum	40
2.4	Boolean matrices: The Four-Russians algorithm	41
2.5	Matrix multiplication and inversion	42
3.1	Interval cover	64
3.2	Memory usage	64
3.3	Scheduling dependent tasks on several machines	65
3.4	Scheduling independent tasks with priorities	66
3.5	Scheduling independent tasks with deadlines	66
3.6	Edge matroids	67
3.7	Huffman code	67
4.1	Matrix chains	87
4.2	The library	88
4.3	Polygon triangulation	88
4.4	Square of ones	89
4.5	The wind band	89
4.6	Ski rental	89
4.7	Building set	90
5.1	Binary counter	108
5.2	Inserting and deleting	108
5.3	Stack	109
5.4	Deleting half the elements	109
5.5	Searching and inserting	109

5.6	Splay trees	110
5.7	Half perimeter of a polygon	112
7.1	Wheel	149
7.2	Knights of the round table	149
7.3	Variants of CLIQUE	149
7.4	Path with vertex pairs	150
7.5	VERTEX-COVER with even degrees	150
7.6	Around 2-PARTITION	150
7.7	COLOR	151
7.8	3-COLOR	151
7.9	3-COLOR-PLAN	152
7.10	Scheduling independent tasks with p processors	152
7.11	Scheduling with two processors	152
7.12	Transitive subchain	153
7.13	INDEPENDENT SET	153
7.14	DOMINATING SET	153
7.15	Carpenter	153
7.16	k -center	153
7.17	Variants of 3-SAT	154
7.18	Variants of SAT	154
7.19	SUBSET-SUM	155
7.20	NP-completeness of 2-PARTITION	155
9.1	Single machine scheduling	211
9.2	SUBSET-SUM	212
9.3	SET-COVER	213
9.4	VERTEX-COVER	213
9.5	Scheduling independent tasks in parallel	215
9.6	Point clustering	215
9.7	k -center	216
9.8	Knapsack	217
9.9	Mixed integer linear program for replica placement	218
9.10	A randomized algorithm for independent set	218
9.11	Branch-and-bound applied to MAX-SAT	219