INDEX

abstract data type (ADT), 1, 43
exercise solutions, 414
functions vs., 43
merging with queue, 56
specifications/implementations, 44
dictionary, 47
graphs, 47
link list implementation, 51
list, 44
orders, 48
priority queue, 46
queue, 45
set, 46
set system, 47
simple types, 44
stack, 44; parsing with stack, 57
trees, 48
AVL tree, 49
binary search tree, 48
recursive definition of tree, 130
union-find set system, 49

Ackermann's function, 127
algorithm, 127
brute force algorithm, 66, 226
defined, 1
Dijkstra's shortest-weighted path algorithm, 183
dynamic programming algorithms and examples, 267
all pairs, matrix multiplication, 314
best AVL tree, 311
chains of matrix multiplications, 306
context-free grammar parsing, 315
dynamic programming algorithms. via reductions, 318
longest-common-sequence problem, 295
longest increasing contiguous subsequence example, 301
longest increasing subsequence example, 301
shortest weight path, directed leveled graph example, 267
weighted job/event scheduling problem, 303
Euclid's greatest common divisor (GCD) algorithm, 79
graph algorithms
expander graphs, 351
max cut problem, 350
minimum spanning tree, 244
network flows, 198

harmonic sum, 395
ratio between terms, 393
solution classes, 389
 technique, 389

ADT. See abstract data type algorithms. See also recursive algorithms

Cambridge University Press
978-0-521-84931-9 - How to Think About Algorithms
Jeff Edmonds

© Cambridge University Press
www.cambridge.org
Index

algorithms (cont.)
  shortest weight path, directed leveled
graph example, 267
3-Colouring, 330, 338
graph search algorithms, 173
  breadth-first search, shortest path, 179, 181
depth-first search, 188
generic search algorithm, 174
  partial order linear ordering, 194; depth-
  first search algorithm, 195; easy but slow
  algorithm, 195
  recursive depth-first search, 192
greedy algorithms, 225, 260, 295, 307
hill-climbing algorithm, 221
iterative algorithms, 8, 12, 21
iterative sorting algorithms, 71, 72, 75, 76
looking forward vs backward, recursive
  algorithms, 99
meta-algorithms, xi, 2
  parsing algorithm specifications, 161
randomized algorithms, 346
  sorting and selection algorithms, 114
steepest-ascent hill-climbing algorithm, 214
Strassen's matrix multiplication, 126
time/space complexity, 82, 85, 324, 347, 366,
  378, 406, 432
analytical sums, 396
arithmetic sums, 394. See also
  adding-made-easy approximations
asymptotic growth, 377
asymptotic growth rate, 378
asymptotic notation, 85, 377, 384
  BigOh definition, 85, 385
  Little Oh definition, 85, 386
  Little Omega definition, 85, 386
loop invariant for lower bounds, 85
  Theta definition, 85, 385
exercise solutions, 432
function classification, 379
growth rates classes, 377
  purpose, 378
AVL tree, 49
  best AVL tree, 311
  best AVL tree problem, 313
binary search 24
  for cost in optimization problems, 329
  narrowing the search space example, 24
  returning index example, 89
  returning yes/no, 91
trees, 48, 60
  ADT, 48
  balanced, 49
best binary search tree problem, 311
  basic steps, 60
coding/implementation details, 61
  ending, 61
establishing loop invariant, 61
  exit condition, 61
is tree a binary tree example, 138
  loop invariant, 60
  main steps, 61
maintain loop invariant, 61
make progress, 61
measure of progress, 60
nodes in binary search tree example, 131
running time, 61
  specifications, 60
typical errors, 26
BigOh definition, 385
bipartite edges, 191
bipartite matching using network flow, 342
breadth-first search, 178
  shortest path, 179, 181
code, 181
  exiting loop, 183
initial code, 183
loop body, 180
loop invariant, 180
maintaining loop invariant, 181
optimization problem, 179
  shortest path problem, 179
  shortest path proof, 180
bucket sort by hand, 71
  basic steps, 71
exit condition, 72
  loop invariant, 72
maintain loop invariant, 72
  specifications, 71
chains of matrix multiplications, 306
failed dynamic programming algorithm, 307
failed greedy algorithm, 307
little bird question, 307
optimal solution construction, 310
optimal solution cost, cost for subinstances,
  308
recursive structure, 308
reduced to subinstance, 308
set of subinstances called, 308
table fill order, 308
table indexed by subinstances construction,
  308
time/space requirements, 311
coloring the plane, 29
  basic steps, 29
coding/implementation details, 30
  ending, 30
establishing loop invariant, 30
Index

exit condition, 30
loop invariant, 29
main steps, 30
maintain loop invariant, 30
measure of progress, 29
running time, 30
special cases, 30
specifications, 29.
computational complexity, 1
asymptotic notations, 85, 377, 384
formal proof of correctness, 408
nondeterministic polynomial-time decision problems (NP), 330
time/space complexity, 82, 85, 324, 347, 366, 378, 406, 432
context-free grammar parsing, 159
abstract data type parsing, 161
correctness proof, 165
dynamic programming example, 315
abstract data type parsing, 316
base cases, 317
help from friend, 316
little bird question, 316
not look ahead one, 315
number of parsings, 316
optimal solution construction, 317
parsing problem, 315
set of subinstances, 317
table fill order, 317
table indexed by subinstances construction, 317
exercise solutions, 421
expression parsing, 160
GetExp code, 163
GetExp expressions, 161
GetExp, Get Term, GetFact examples, 162
GetFact code, 164
GetTerm code, 163
grammar, 159
look ahead one, 159, 165
nonterminals, 160
parsing algorithm specifications, 161
rules, 160
running time, 165
semantics and, 159
syntax and, 159
terminals, 160
tree of stack frames, 165
correctness, 2
formal proof of correctness, 408
for context-free grammar parsing, 165
for depth-first search, 196
for dynamic programming, 283
for greedy algorithms, 230
for recursive algorithms, by strong induction, 113
counting sort, 72
basic steps, 73
code, 74
establishing loop invariant, 74
exit condition, 74
loop invariant, 74
main step, 74
maintain loop invariant, 74
running time, 75
 specifications, 72
cyclic edges, 191
Davis-Putnam, 263
depth-first search, 178, 188
code, 189
depth-first search, 189
dependencies, 189
decomposing search, 189
deterministic finite automation (DFA), 31
addition example, 35
applications, 31
calculator example, 36
compiling iterative program into DFA, 33
division example, 36
dynamic programming, 38
longest block of ones example, 37
longest increasing contiguous subsequence, 38
longest increasing subsequence, 38
Dijkstra's shortest-weighted-path algorithm, 183
code, 186
exiting loop, 188
initial code, 187
loop body, 186
loop invariant, 185
maintaining L11, 186
maintaining L12, 187
problem specifications, 183
shortest distance approximation, 184
shortest path proof, 184
dynamic programming, 267
exercise solutions, 426, 427
steps in developing, 267
base cases, 274
code, 275
count subinstances, 273
final solution, 275
redundancy, 272
running time, 277
set of subinstances, 272
Index

dynamic programming (cont.)
solution from subsolutions, 274
table fill order, 275
table indexed by subinstances
construction, 273
recursive backtracking, 267
running time, 271
subtle points, 277
dynamic programming algorithms and
examples
all pairs, matrix multiplication, 314
best AVL tree, 311
best AVL tree problem, 313
best binary search tree, 311
chains of matrix multiplications, 306
context-free grammar parsing, 315
dynamic programming algorithms via
reductions, 318
best path similarity, 319
bigger-is-smarter elephant problem, 322
event scheduling problem, 319
graph instance formation, 319
mapping back algorithm, 321
longest-common-sequence problem, 295
base cases, 297
code, 297
greedy algorithm, 295
information about subinstance, 299
little bird possible answers, 295
little bird question, 296
longest common sequence, 295
optimal solution construction, 299
set of subinstances, 296
table fill order, 297
table indexed by subindexes construction, 297
time/space requirements, 299
longest increasing contiguous subsequence
example, 301
longest increasing subsequence example, 301
time/space requirements, 300
shortest weight path, directed leveled graph
example, 267
weighted job/event scheduling problem, 303
failed algorithms, 303
greedy dynamic programming, 304

Euclid’s GCD algorithm. See greatest common
divisor algorithm
Euler cycle, 40
event scheduling 236, 319
weighted job/event scheduling problem, 303
existential/universal quantifiers, 357, 358, 372
bound variables definition, 359
combining quantifiers, 359
exercise solutions, 431
expressions building, 360
free variables definition, 359
Loves example, 358
negation, 361
quantifiers order, 359
relation definition, 358
representations, 358
variable domain, 360
exponentials, 374
base, 376
exercise solutions, 432
ratio, 376
rules, 375
uses, 374
fast Fourier transformation, 125
find-max two-finger algorithm example, 10
forward/cross edges, 191
friends level of abstraction, recursive
algorithms, 100
base cases, 101
general input, 100
generalizing problem, 101
link to techniques, iterative algorithms, 102
minimizing number of cases, 101
running time, 102
size, 100
specifications, 100
functions. See also Ackermann’s function
abstract data types vs., 43
analytical functions, simple, 396
linear function, 382
quadratic function, 382
time/space complexity as, 367

GCD algorithm. See greatest common divisor
(GCD) algorithm
grouped sums, simple, 393
GetExp code, 163
GetExp expressions, 161
GetExp, GetTerm, GetFact examples, 162
GetExp reasoning, 163
GetFact code, 164
GetFact reasoning, 164
GetTerm code, 163

edges classification, 190
back edges, 190
bipartite edges, 191
cross edges, 191
cyclic edges, 191
forward edges, 191
tree edges, 190
global vs. local considerations, 200, 204, 222, 226, 228, 252, 278

graph algorithms
expander graphs, 351
max cut problem, 350
minimum spanning tree, 244
network flows, 198
shortest weight path, directed leveled graph example, 267
3-Colouring, 330, 338
graph search algorithms, 173
breadth-first search, shortest path, 179, 181
depth-first search, 188
Dijkstra's shortest-weighted-path algorithm, 183
exercise solutions, 422

generic search algorithm, 174
basic steps, 174
code, 175
exit condition, 176
exiting loop, 177
handling nodes order, 178
initial code, 176
loop body, 175
loop invariant, 174
maintaining loop invariant, 176
measure of progress, 176
reachability problem, 174
running time, 177
partial order linear ordering, 194
recursive depth-first search, 192

graph theory problems, 173

greatest common divisor (GCD) algorithm, 79
code, 81
ending, 79, 81
establishing loop invariant, 79
example, 82
exercise solutions, 415
exit condition, 81
iteration on general instance, 79
loop invariant, 79
lower bound, 82
making progress, 80, 80
recursive, 126
running time, 82
special cases, 80
specifications, 79
termination, 81
greedy algorithms, 225
brute force algorithm, 226
correctness proof, 230
using loop invariants, 227
examples
game show, 226
interval cover problem, 240
job/event scheduling problem, 236
minimum spanning tree problem, 244
exercise solutions, 423
fixity vs. adaptive priority, 234
greedy choice, 226
loop invariants, types, 227
specifications, 225

harmonic sum, 395
close to harmonic, 395
heap sort/priority queues, 141
array implementation, balanced binary tree, 141
common mistakes, 147
completely balanced binary tree, 141
heap definition, 141
heapify problem, 142
code, 143
iterative algorithm, 143
recursive algorithm, 142
running time, 143, 144
specifications, 142

heapsort problem, 145
algorithm, 145
array implementation, 146
code, 147
specification, 145

makeheap problem, 144
iterative algorithm, 144
recursive algorithms, 144
running time, 145
specifications, 144

priority queues, 147

hill-climbing algorithm, 221. See also
primal-dual hill climbing method:
steepest-ascent hill-climbing algorithm

small local maximum, 200
algorithm, faulty, 203
algorithm fixing, 205
augmentation graph, faulty, 202
basic ideas, 201
counterexample, 204
local maximum, 204

image drawing see recursive image drawing
information hiding, 43
information theoretic lower bounds, 87, 92

iterative algorithms, 8, 12
basic steps, 13
code structure, 8
coding/implementation details, 19
correctness proof, 8
ending, 19
establishing loop invariant, 17
exit condition, 18
Index

iterative algorithms (cont.)
find-max two finger algorithm example, 10
formal proof, 20
loop invariants for, 8, 13, 15
main steps, 16
maintain loop invariant, 16
make progress, 16
measure of progress, 13
running time, 8, 19
special cases, 19
specification, 12
types of iterative algorithms, 21
case analysis, 22
more of input, 21; insertion sort example, 23
more of output, 21; selection sort example, 22
search space narrowing, 22
binary search narrowing example, 24
work done, 22
bubble sort example, 25
iterative sorting algorithms
bucket sort by hand, 71
counting sort, 72
radix counting sort, 76
radix sort, 75

job/event scheduling see event scheduling

kth smallest element example, 117

Las Vegas model, 347
linear function, 382
linear programming, 219 see also network flow
   Euclidean space interpretation, 220
   example, 219
formal specification, 219
hill-climbing algorithm, 221
matrix representation, 220
network flows, 220
running time, 223
small local maximum, 222
link list implementation, 51
   adding node to end, 53
   adding node to front, 51
   deleting node, 55
   hidden invariants, 51
   initialize walk, 54
   notation, 51
   removing node from end, 53
   removing node from front, 52
testing whether empty, 53
   walking down linked list, 53
Little O definition, 85, 386
Little Omega definition, 85, 386
logarithms, 374
   base, 376
   exercise solutions, 432
   ratio, 376
   rules, 375
   uses, 374
longest block of ones example, 37
longest-common-sequence problem, 169, 295
   base cases, 297
code, 297
greedy algorithm, 295
little bird possible answers, 295
little bird question, 296
longest common sequence, 295
optimal solution construction, 299
set of subinstances, 296
table fill order, 297
table indexed by subindexes construction, 297
time/space requirements, 299
longest increasing contiguous subsequence example, 301
longest increasing subsequence example, 301
looking forward to; backward, recursive algorithms, 99
algorithm, 99
loop invariant for lower bounds, 85
asympotic notation, 85
binary search returning index example, 89
binary search returning yes/no, 91
dynamic algorithms, 90
exercise solutions, 415
flipping a bit, 90
loop invariant argument, 86
lower bounds proof, state of art, 92
multiplexer example, 90
parity example, 89, 90
sorting example, 87
time complexity, 85
upper bound, algorithm, 85
loop invariants for iterative algorithms, 8, 13, 15
lower bounds, 82, 85, 92, 264, 325, 415, 425 See also loop invariant for lower bounds
   for GCD, 82
loop invariant for lower bounds, 85
lower bounds proof, state of art, 92
reductions, 324

magic sevens, 62
   basic steps, 63
   establishing loop invariant, 64
exit condition, 64
loop invariant, 63
maintain loop invariant, 63
Index

running time, 64
deterministic finite automaton, 31
in dynamic programming
longest increasing contiguous subsequence example, 301
longest increasing subsequence example, 301
in greedy algorithms, 228
more-of-the-input vs. more-of-the-output, 39
tournament example, 39
recursive algorithms, link to, 102
in VLSI chip testing example, 67
more-of-the-output loop invariant algorithms, 21
selection sort example, 22
Euler cycle example, 40
multiplexer example, 90, 90

classifying problems, 326
NP completeness proof steps, 330, 331
nondeterministic polynomial-time decision problems (NP), 330
reduction P_{	ext{poly}} \rightarrow P_2, 324
reverse reductions, 325
satisfiability vs. optimization

Algorithms, 171
examples, 172
airplane, 172
course scheduling, 172
longest common sequence, 172
network flow, 198
problem specification, 171

operations on integers, 122

b^N example, 122
Strassen's matrix multiplication example, 126
.xyz example, 123

optimization problems, 171
examples, 172
airplane, 172
course scheduling, 172
longest common sequence, 172
network flow, 198
problem specification, 171

parity example, 89, 90
parsing with stack, 57. See also context-free parsing
code, 58
ending, 58
disexample, 57
initial conditions, 58
loop invariant, 58
maintaining loop invariant, 58
parsing only, 59
parsing with context-free grammar, 59
specifications, 57

partial order linear ordering, 194
depth-first search algorithm, 195
easy but slow algorithm, 195
partial order definition, 194
shortest weight path, DAG, 196
topological sort problem specifications, 195
total order definition, 194
postconditions, 1
preconditions, 1
primal-dual hill climbing method, 206
algorithm, 206
ending, 209
max-flow-min-cut duality principle, 213
running time, 213
Index

primality testing, randomized, 349
printing neatly example, 277

quantifiers. See existential/universal quantifiers
quadratic function, 382
quick sort
  example, 116
  randomized, 348

radix counting sort, 76
  algorithm, 77
  example, 77
  running time, 78
  specifications, 77
radix sort, 75
  basic steps, 75
  ending, 76
  establishing loop invariant, 76
  loop invariant, 76
  maintain loop invariant, 76
  specification, 75
randomized algorithms, 346
  hiding worst cases from adversary, 346, 347
  deterministic worst case model, 347
  game show problem, 348
  Las Vegas model, 34
  Monte Carlo model, 348
  quick sort, 348
  randomized counting, 349
  randomized primality testing, 349
  optimization problems with random
    structure, 350
    expander graphs, 351
    max cut problem, 350
  VLSI chip testing, 69
randomly generating maze, 156
  initial conditions, 157
  postcondition, 156
  precondition, 156
  running time, 158
  searching maze, 158
  subinstances, 157
recurrence relations, 398
  exercise solutions, 433
  proofs, 401
  in recursive backtracking algorithms, 259
  recursive programs timing, 398
  solving recursive relations, 399
recursion 97
  friends level of abstraction, 100
  on trees
    exercise solutions, 419
    generalizing problem solved, 138
    heap sort/priority queues, 141
  recursive definition of tree, 130
  representing expressions with trees, 149
  simple examples, 135
recursive algorithms see also recursive
  backtracking
  checklist for, 104
  code structure, 104
  specifications, 105
  tasks to complete, 107
  variables, 105
correctness proof, with strong induction, 113
  examples
  Ackermann’s function, 127
  exercise solutions, 417
  operations on integers, 122
  \(b^n\) example, 122
  Strassen’s matrix multiplication example, 126
  \(xyz\) example, 123
exercise solutions, 416
  base cases, 101
  general input, 100
  generalizing the problem, 101
  link to techniques, iterative algorithms, 102
  minimizing number of cases, 101
  running time, 102
  size, 100
  specifications, 100
looking forward \(\text{vs.}\) backward, 99
solving, 127
sorting/selecting algorithms, 114
  choosing the pivot, 118
  finding the \(k\)th smallest example, 117
  general recursive sorting algorithm, 114
  merge sort example, 114
  partitioning according to pivot element, 120
  quick sort example, 116
  stack frame, 110
  strong induction, 112, 112
  tower of Hanoi, 102
recursive backtracking algorithms, 251, 425
  as sequence of decisions, 251
  best animal searching example, 253
  maze searching example, 252
  developing steps, 256
  exercise solutions, 425
  pruning branches, 260
  greedy algorithms, 260
  queens problem example, 256
satisfiability, 261
  code, 264
  Davis–Putnam, 263
  instances/subinstances, 262
  pruning, 263
Index

running time, 264
satisfiability problem, 261
solutions iterating, 262
recursive depth-first search, 192
achieving postcondition, 193
code, 192
eexample, 193
running time, 194
recursive images, 153
exercise solutions, 421
fixed recursive/base case image, 153
base case, 153
birthday cake, 154
eexamples, 154
fractal, 155
image drawing, 153
man recursively framed, 154
recursing, 154
rotating square, 154
randomly generating maze, 156
recursive image specification, 153
reduction, 324
dynamic programming algorithms via, 318
best path similarity, 319
bigger-is-smarter elephant problem, 322
event scheduling problem, 319
graph instance formation, 319
mapping back algorithm, 321
lower bound, 325
optimization problems reduction, 326
reverse reduction, 325
upper bound, 324
use in classifying problems, 326
running time, 2. See also time/space complexity
Ackermann's function, 128
best binary search tree, 313
binary search, 24
binary search trees, 61
coloring the plane, 30
context-free grammar parsing, 165
counting sort, 75
dynamic programming, 277
GCD algorithm, 82
generic search algorithm, 177
heapsort, 147
heapify, 143
interval cover, 243
iterative algorithms, 8, 19
job/event scheduling, 240
linear programming, 223
makeheap, 145
magic sevens, 62
merge sort, 115
minimum spanning tree, 247
nodes in binary tree, 132
queens problem, 260
quick sort, 116
primal-dual hill climbing method, 213
radix counting sort, 78
randomly generating maze, 158
recursive algorithms, 102
recursive backtracking, 271
satisfiability, 264
steepest-ascent hill-climbing algorithm, 216
towers of Hanoi, 104
union find set system, 50
VLSI chip testing, 69
shortest weight path, directed leveled graph example, 267
sorting and selection algorithms, 114
bubble sort example, 25
finding the 4th smallest example, 117
choosing the pivot, 118
iterative sorting algorithms
bucket sort by hand, 71
counting sort, 72
radix counting sort, 76
radix sort, 75
merge sort, 115
quick sort, 116
randomized quick sort, 348
recursive sorting algorithm, general, 114
stack frame, 110
memory, 111
stack of stack frames, 110
tree of stack frames, 98, 110, 132, 165
using, 111
steepest-ascent hill-climbing algorithm, 214
augmentation path, 215
running time, 216
Strassen's matrix multiplication example, 126
time/space complexity, 82, 82, 85, 324, 347, 366, 378, 406, 432
examples, 369, 369
exercise solutions, 432
formal definition, 371
as functions, 367
operation definition, 368
purpose, 366
size definition, 367
time complexity of problem, 368
tournament example, 39
towers of Hanoi, 102
code, 104
Index

towers of Hanoi (cont.)
  divide and conquer, 103
  running time, 104
  specification, 102, 103
  subinstance, 104

tree edges, 190

trees, 48. See also binary search trees; recursion
  on trees
  AVL tree, 49
  best AVL tree problem, 313
  best binary search tree, 311
  binary search tree, 48
  chains of matrix multiplications, solution as
  tree example, 306
  is a tree a binary search tree example, 138
  nodes in binary tree example, 131
representing expressions with
  differentiate expression example, 150
  evaluate expression example, 149
  recursive definition of expression, 149
  simplify expression example, 151
  tree data structure, 149
recursive definition of, 130
of stack frames, 98, 110, 132, 165
traversals, 133

universal quantifiers. See existential/universal
  quantifiers
  upper bound, 82
  cuts as upper bound, 209
  for hill-climbing, 222

VLSI chip testing, 65
  brute force algorithm, 66
  data structure, 66
  exiting loop, 69
  extending the algorithm, 69
  faster algorithm, 67
  initial code, 69
  loop invariant design, 67
  maintaining loop invariant, 68
  measure of progress, 68
  randomized algorithm, 69
  running time, 69
  specification, 65

weighted job/event scheduling problem, 303
  failed algorithms, 303
  greedy dynamic programming, 304
  weighted event scheduling problem, 303
  work done-bubble sort example, 25