CAMBRIDGE
Cambridge University Press & Assessment
978-0-999-28298-4 — Algorithms Illuminated
Tim Roughgarden
Index
More Information

Index

$|x|$ (absolute value), 35
$\binom{n}{2}$ (binomial coefficient), 33, 53, 146
$\lceil x \rceil$ (ceiling), 134
$\vee$ (disjunction), 542
$n!$ (factorial), 115, 293, 443, 519
$\lfloor x \rfloor$ (floor), 68
$\neg$ (logical negation), 542
$|S|$ (set size), 144
$=$ vs. $:=$, 7
$1 - \frac{1}{e}$, 486
2-OPT algorithm, 500–501
    2-change, 499
    implementation, 517, 518, 649
    improving 2-change, 499
    is interruptible, 501
    pseudocode, 500
    running time, 501
    solution quality, 501
    vs. 3-OPT, 506, 509
2-SAT, 197, 548
2-change, *see* 2-OPT algorithm
2-SUM, 260–262, 289
3-SAT
    and the Exponential Time Hypothesis, 587
    is NP-complete, *see* Cook-Levin theorem, stronger version
    is NP-hard, *see* Cook-Levin theorem
    padded, 595
    problem definition, 553
    reduces to directed Hamiltonian path, 563–567
    reduces to graph coloring, 576, 653
    reduces to independent set, 559–562
    reduction from an arbitrary $\mathcal{NP}$ problem, 582–584
    Schöning's algorithm, 548–550, 652
3-SUM, 287
63.2%, 486

A* search, 212–213, 263
Aaronson, Scott, 586
Aarts, Emile, 508
abstract data type, 214
Ackermann function, 350
acknowledgments, xvi
ACM, 39
adjacency lists, 147–148
    in graph search, 161
    input size, 150
    vs. adjacency matrix, 149
adjacency matrix, 148–149
    applications, 149
    input size, 150
    sparse representation, 150
    vs. adjacency lists, 149
Adleman, Leonard, 132
Aho, Alfred V., 5
Albertini, Ange, 543
algorithm, 2
    anytime, 501
    approximation, 456, 474
    constant-time, 28
    design paradigms, 46
    exponential-time, 449
    fast, 23, 445
    fixed-parameter, 534, 588
    heuristic, 456
    linear-time, 23
    mind-blowing, 45
    online, 477
    polynomial-time, 448
    pseudopolynomial-time, 570
    quadratic-time, 28
    quantum, 452
    randomized, 452
    subexponential-time, 453
algorithm design paradigm, 291
    divide-and-conquer, *see* divide-and-conquer algorithms
    dynamic programming, *see* dynamic programming
    greedy algorithms, *see* greedy algorithms
    local search, *see* local search
algorithm field guide, 630–631
algorithmic game theory, 596
all-pairs shortest path problem, *see* shortest paths, all-pairs
Alon, Noga, 527
alphabet, *see* code, alphabet
among friends, 11, 14, 72, 299, 340, 403
Applegate, David L., 445
applications, 2

©2022, Soundlikeyourself Publishing, LLC

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-0-999-28298-4 — Algorithms Illuminated
Tim Roughgarden
Index
More Information

approximation algorithm, 456, 474
approximation ratio, 474
    as an insurance policy, 475
Aquarius Records, 191
arc, *see* edge (of a graph), directed
argmax, 316
argmin, 316
asymptotic analysis, 22
asymptotic notation, 27–42
    as a sweet spot, 27
    big-O notation, *see* big-O notation
    big-O vs. big-theta notation, 38
    big-omega notation, 37
    big-theta notation, 38
    history, 39
    in seven words, 27
    little-o notation, 39
auction, *see* FCC Incentive Auction
`Augmented-BFS`, 164
average-case analysis, 20

backtracking, 536
Backurs, Arturs, 589
Bacon number, 153, 164
Bacon, Kevin, 154
base case (induction), 617
base case (recursion), 7
beam search, 508, *see also* local search
Bellman, Richard E., 380, 418, 524
Bellman-Ford algorithm, 418–429
    and Internet routing, 427
    correctness, 423–424
    example, 424–426
    optimal substructure, 420–421
    pseudocode, 423
    reconstruction, 427
    recurrence, 422
    running time, 426–429
    space usage, 427
    stopping criterion, 422
    subproblems, 419–420
Bellman-Held-Karp algorithm (for the TSP), 519–525
    correctness, 524
    example, 545, 649
    implementation, 550
    memory requirements, 522, 547
    optimal substructure, 521–522, 524–525
    pseudocode, 524
    reconstruction, 524, 547
    recurrence, 522
    running time, 524
    subproblems, 523
    variations, 546
BFS, *see* breadth-first search
`BFS`, 160
Biere, Armin, 543
big-O notation, 33–35
    as a game, 34
    English definition, 33
    high-level idea, 28
    in an exponent, 581
    mathematical definition, 34
    pictorial definition, 33
big-omega notation, 37
big-picture analysis, 21
big-theta notation, 38
binary search, 75, 238, 242
binary search tree, *see* search tree
bipartite graph, 196, 548
birthday paradox, 265, 272–273, 628
bit, 113, 306
Bixby, Robert E., 445
blazingly fast, xiii, xiv, 23, 65, 90, 91, 97, 153, 474, 607
bloom filter
    INSERT, 277, 279
    LOOKUP, 277, 280
    applications, 278–279
    has false positives, 277, 280
    has no false negatives, 280
    heuristic analysis, 282–285
    heuristic assumptions, 282
    in network routers, 279
    operation running times, 278
    raison d'être, 277
    scorecard, 278
    space-accuracy trade-off, 278, 281, 285, 287
    supported operations, 277
    vs. hash tables, 277–278
    when to use, 278
Bloom, Burton H., 277
Blum, Manuel, 132
Boolean, 542
Borodin, Allan, 291
bow tie, *see* Web graph
branch and bound, 536, *see also* MIP solvers
breadth-first search, 157–163, 330, 349
    and bipartite graphs, 196
    correctness, 163
    example, 161–162
    for computing connected components, 166–168
    for computing shortest paths, 164–165
    layers, 159, 165
    pseudocode, 160
    running time analysis, 163
Broder, Andrei, 191
broken clock, 204
brute-force search, *see* exhaustive search
`BubbleSort`, 11, 112, 640
`BucketSort`, 113

©2022, Soundlikeyourself Publishing, LLC

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-0-999-28298-4 — Algorithms Illuminated
Tim Roughgarden
Index
More Information

Bursztein, Elie, 543

C++, 238, 240
cache, 373
can we do better?, 5, 209, 306, 335, 444, 451, 477, 576, 577, 589
cascade model, 490
Cayley's formula, 330
cf., 83
Chen, Ke, 635
chess, 262
`ChoosePivot`
  median-of-three implementation, 118
  naive implementation, 98, 100
  overkill implementation, 99, 101
  randomized implementation, 102
Chvátal, Vašek, 445
clause, 542, *see also* constraint
Clay Mathematics Institute, 586
clique problem
  is NP-hard, 575, 652
  reduction from independent set, 575, 652
closest pair
  correctness, 65–67
  exhaustive search, 59
  one-dimensional case, 60
  problem definition, 59
  pseudocode, 60, 62
  running time, 62
clustering, 167, 358–361
  k-means, 360
  and Kruskal's algorithm, 361
  choosing the number of clusters, 359
  greedy criterion, 360
  informal goal, 359
  similarity function, 359
  single-link, 361
Cobham, Alan, 449
cocktail party, xv, 215, 272, 447
code
  Σ-tree, 312
  alphabet, 306
  as a tree, 309–312
  average leaf depth, 312
  average encoding length, 308
  binary, 306
  encodings as root-leaf paths, 311
  fixed-length, 306
  Huffman, *see* Huffman's algorithm
  optimal prefix-free, 308, 313
  prefix-free, 307, 312
  symbol frequencies, 308, 309
  ternary, 327
  variable-length, 306
coin flipping, 127, 275
collaborative filtering, 47
collision, *see* hash function, collision
color coding, 525–535
  and minimum-cost panchromatic paths, 529–531
  correctness, 530, 534
  example, 546, 650
  in practice, 535
  minimum-cost k-path problem, 526
  motivation, 525
  panchromatic path, 528
  pseudocode, 533
  reconstruction, 547
  recurrence, 529
  running time, 531, 533
  subproblems, 529
  with random colors, 531
compression, 306
compromising
  from day one, 578
  on correctness, 454–456, 471–518
  on generality, 454–455, 470, 548
  on speed, 454, 456–457, 519–550
computational complexity theory, 577–593
  fine-grained, 589
computational genomics, 445, 525, *see also* sequence alignment
computational geometry, 59
computational lens, 3
computationally intractable, *see* NP-hardness
conflict-driven clause learning, 536, *see also* SAT solvers
connected components
  applications, 167–168
  definition, 166
  example, 168
  in directed graphs, *see* strongly connected components
  linear-time computation, 168–170
  number of, 170
constant, 35, 74
  reverse engineering, 36, 137
constant factors, 21, 28
constraint
  in mixed integer programming, 538
  in satisfiability, 542
Cook reduction, 578
Cook, Stephen A., 553, 578
Cook, William J., 445
Cook-Levin theorem, 553–554
  50th anniversary, 553
  formal statement, 582
  history, 553
  proof sketch, 582–584
  stronger version, 592
Cormen, Thomas H., 252
Cornuéjols, Gérard P., 484

©2022, Soundlikeyourself Publishing, LLC

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-0-999-28298-4 — Algorithms Illuminated
Tim Roughgarden
Index
More Information

corollary, 15
counting inversions
  correctness, 49, 52
  exhaustive search, 47
  implementation, 70
  problem definition, 46
  pseudocode, 48, 49
  running time, 52
  split inversions, 51
`CountingSort`, 113
  stable implementation, 113
Coursera, xvi
coverage, 481, *see also* maximum coverage
  is submodular, 515
CPLEX, *see* MIP solvers
Crosby, Scott A., 270
cryptography, 276
culturally acceptable inaccuracies, 466
cut (of a graph), 364
Cut Property, *see* minimum spanning tree, Cut Property
cycle (of a graph), 329
  negative, 417
Cycle Property, *see* minimum spanning tree, Cycle Property
cycle-free shortest paths
  as an optimization problem, 579
  is NP-hard, 462, 555
  problem definition, 462
  reduction from directed Hamiltonian path, 463
Cygan, Marek, 534

DAG, *see* directed acyclic graph
Dasgupta, Sanjoy, 73
data structure
  bloom filter, *see* bloom filter
  deft deployment, 630
  disjoint-set, *see* union-find
  expertise levels, 215
  hash table, *see* hash table
  heap, *see* heap, *see* heap
  principle of parsimony, 215
  queue, 160, 214, 319
  raison d'être, 214
  scorecards, *see* scorecards
  search tree, *see* search tree
  stack, 172, 214
  union-find, *see* union-find
  vs. abstract data type, 214
de-duplication, 26, 260
decision problem, 579
decomposition blueprint, 106, 127, 274
degree (of a vertex in a graph), 151, 421
depth (of a node in a tree), 311
depth-first search, 158, 170–174, 330, 349
  correctness, 173
  example, 170
  for computing connected components, 174
  for computing strongly connected components, 183
  for topological sorting, 177–180
  iterative implementation, 172
  recursive implementation, 173
  running time analysis, 174
derandomization, 453
descending clock auction, *see* FCC Incentive Auction
design patterns, xiv
DFS, *see* depth-first search
`DFS` (Iterative Version), 172
`DFS` (Recursive Version), 173
`DFS-SCC`, 187
`DFS-Topo`, 178
diameter (of a graph), 194, 595
dictionary, *see* hash table
`diff`, 413
`Dijkstra`, 202
`Dijkstra` (heap-based), 224, 225
Dijkstra's shortest-path algorithm
  and A* search, 212–213, 263
  as a greedy algorithm, 292
  correctness, 206–208, 299
  Dijkstra score, 203
  example, 204
  for computing minimum bottleneck paths, 212, 236
  greedy selection rule, 203
  heap-based implementation, 222–226
  in undirected graphs, 200
  pseudocode, 202
  pseudocode (heap-based), 224, 225
  reconstructing shortest paths, 203
  resembles Prim's algorithm, 331
  running time analysis, 209
  running time analysis (heap-based), 225
  straightforward implementation, 209
  with negative edge lengths, 205, 416
Dijkstra, Edsger W., 198, 331
directed acyclic graph, 176
  has a source vertex, 176
  has a topological ordering, 176–177
discussion forum, xvi
disjunction (of literals), 542
dist, *see* shortest paths, distance
distance, *see* shortest paths, distance
divide-and-conquer, 9, 10, 45–46, 291–292, 630
  for closest pair, 60
  for counting inversions, 48
  for matrix multiplication, 55
  for sorting, 45
  proofs of correctness, 93

©2022, Soundlikeyourself Publishing, LLC

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-0-999-28298-4 — Algorithms Illuminated
Tim Roughgarden
Index
More Information

vs. dynamic programming, 375, 379–380
when to use, 46
double summation, 626
Draper, Don, 154
DSelect
30-70 Lemma, 133–135
as a knockout tournament, 129
does not run in place, 131
heuristic analysis, 136
history, 132
pseudocode, 129
running time, 132
running time analysis, 132–137
vs. RSelect, 129, 132
with groups of 3 or 7, 139
Dumitrescu, Adrian, 635
dynamic programming, xiii
as recursion with a cache, 372
bottom-up, 373
for all-pairs shortest paths, *see* Floyd-Warshall algorithm
for beating exhaustive search, 519–525, 529–531, 631
for color coding, 529–531
for graph problems, 419
for knapsack, 455, 469, 515, 649
for optimal binary search trees, 400–410
for single-source shortest paths, *see* Bellman-Ford algorithm
for the knapsack problem, 380–387
for the sequence alignment problem, 392–399
for the TSP, *see* Bellman-Held-Karp algorithm
for weighted independent set in path graphs, 370–377
history, 380
memoization, 373
optimal substructure, 378
ordering the input, 382
principles, 377–378, 520
recurrence, 379
running time, 378, 520
saving space, 412, 645
subproblems, 378–379, 521
takes practice, 366
top-down, 372
vs. divide-and-conquer, 375, 379–380
when to use, 630

e (Euler's number), 486, 519
e.g., 347
Easley, David, 192
edge (of a graph), 142
directed, 143
length, 198, 415
parallel, 144, 148, 330
undirected, 143
weighted, 148
Edmonds, Jack, 445, 449, 577
EdX, xvi
Egoyan, Atom, 154
Einstein, Albert, 214, 578
endpoints (of an edge), 143
equivalence class, 166
equivalence relation, 166
Erdös number, 154
Erdös, Paul, 154
ETH, *see* Exponential Time Hypothesis
Euclidean distance, 59, 518
event (in probability), 621
exchange argument, 299
for minimum spanning trees, 345
in Huffman's algorithm, 321
in scheduling, 300
exhaustive search, 294, 309, 335, 368, 394, 410, 581, 630
for closest pair, 59
for counting inversions, 47
expectation (of a random variable), 492, 494, 623
linearity of, 624
expected value, *see* expectation
Exponential Time Hypothesis
and fixed-parameter algorithms, 588
and Schöning's algorithm, 550, 587
definition, 587
is false for unnatural problems, 595, 653
vs. the P $\neq$ NP conjecture, 453, 587
vs. the SETH, 588

factoring, 453, 465
Fano, Robert M., 313
fast algorithm, 23, 445
Fast Fourier Transform, 69
FCC Incentive Auction
algorithm portfolio, 608
and graph coloring, 603
and greedy heuristic algorithms, 600–604
and SAT solvers, 604–609
and timeouts, 609
and weighted independent set, 599
as a descending clock auction, 609–613
computing payments, 612, 615, 654
feasibility checking, 604–609
final outcome, 613
forward auction, 598
incentives, 613, 615, 654
matches supply and demand, 614
motivation, 596–598
preprocessing, 607
presolvers, 606
repacking problem, 605
representative instances, 602

©2022, Soundlikeyourself Publishing, LLC

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-0-999-28298-4 — Algorithms Illuminated
Tim Roughgarden
Index
More Information

reverse auction, 598
side constraints, 605
station-specific multipliers, 601–602
feasible solution
in local search, 505
to an $\mathcal{NP}$ problem, 579
Federer, Roger, 132, 610
Feige, Uriel, 487
Fermat's Last Theorem, 586
Fibonacci numbers, 389
field guide (to algorithm design), 630–631
Firth, Colin, 154
Fisher, Marshall L., 484
fixed-parameter algorithm, 534, 588
Floyd, Robert W., 132, 430
Floyd-Warshall algorithm, 430–437
detecting a negative cycle, 436–437
optimal substructure, 432–434
pseudocode, 434
reconstruction, 437
recurrence, 434
running time, 436
space usage, 436
subproblems, 430–432
Fomin, Fedor V., 534
for-free primitive, 23, 103, 155, 630
Ford Jr., Lester R., 418
forest (of trees), 315
Fortnow, Lance, 586
Four Color Theorem, 541
Fourier matrix, 69
fully polynomial-time approximation scheme (FPTAS), 516

Gödel, Kurt, 586
Gabow, Harold N., 638
Garey, Michael R., 557, 593
Gauss's trick, 8, 56
Gauss, Carl Friedrich, 8
`GenericSearch`, 155
genetic algorithms, 508, *see also* local search
geometric series, 84–85, 128, 489
Git, 413
golden ratio, 389
good vs. evil, 82
Google, 3, 149
googol, 28, 262, 569, 576
Gosper's hack, 524
gradient descent, 504
Graham's algorithm, 473
approximate correctness, 474–477
bad example, 474, 479
intuition, 475
running time, 474, 514, 648
with small jobs, 511, 647
Graham, Ronald L., 473
graph, 142
k-colorable, 541
adjacency lists, 147–148, 150
adjacency matrix, 148–150, 435
applications, 143–144
bipartite, 196, 548
co-authorship, 154
complete, 146, 166, 443
connected, 144, 443
connected components, *see* connected components
cut, 364
cycle, 175, 329
dense, 145, 430
diameter, 194, 595
directed, 143
directed acyclic, *see* directed acyclic graph
independent set, 366
input size, 144
notation, 142, 144
number of edges, 348
path, 146, 155, 166, 329, 368
planar, 541
radius, 194
representations, 146–149
search, 330, 349
spanning tree, 329, 443
sparse, 145, 429
tour, 443
tree, 146
Web, *see* Web graph
graph coloring, 541
and the FCC Incentive Auction, 603
applications, 541
as satisfiability, 542–543, 604
is NP-hard for $k \geq 3$, 576, 653
problem definition, 541
reduction from 3-SAT, 576, 653
with $k = 2$ (is linear-time solvable), 548
graph isomorphism, 453
graph search
A*, 212–213, 263
applications, 153–155
breadth-first search, *see* breadth-first search
depth-first search, *see* depth-first search
for planning, 154
generic algorithm, 155–158
in game playing, 263
problem definition, 155
greatest hits, xv
greedy algorithm, 291–293
and brainstorming, 368, 473, 630
as a heuristic algorithm, 293, 473, 631
exchange argument, *see* exchange argument
for clustering, 360
for influence maximization, 493

©2022, Soundlikeyourself Publishing, LLC

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-0-999-28298-4 — Algorithms Illuminated
Tim Roughgarden
Index
More Information

for knapsack, 512, 648
for makespan minimization, 473, 477
for maximum coverage, 484
for optimal prefix-free codes, *see* Huffman's algorithm
for scheduling, 295–298
for set cover, 511, 647
for submodular function maximization, 514
for vertex cover, 513, 648
for weighted independent set, 600–602, 616, 654
in the FCC Incentive Auction, 600–604
informal definition, 291, 473
Kruskal's algorithm, *see* Kruskal's algorithm
Prim's algorithm, *see* Prim's algorithm
proof of correctness, 299
themes, 292
usually not correct, 292, 369
`GreedyRatio`, *see* scheduling, `GreedyRatio`
guess-and-check method, 137
guiding principles, 20–23
Gurobi Optimizer, *see* MIP solvers
Gusfield, Dan, 540

Hüffner, Falk, 535
Hadamard matrix, 69
hall of fame, 90, 346
halting problem, 453, 578
Hamiltonian path (directed)
equivalent to undirected Hamiltonian path, 576, 652
example, 463
is NP-hard, 563
problem definition (decision), 462
problem definition (search), 562
reduces to cycle-free shortest paths, 463
reduction from 3-SAT, 563–567
search vs. decision, 562, 575, 652
Hamiltonian path (undirected)
equivalent to directed Hamiltonian path, 576, 652
is NP-hard, 568
problem definition, 567
reduces to the TSP, 568–569
Hamm, Jon, 154
Hart, Peter E., 212
hash function
and the birthday paradox, 266
bad, 269
collision, 265
collisions are inevitable, 265, 270
cryptographic, 276, 543
definition, 264
desiderata, 271
don't design your own, 276
example, 271–272
good, 271
how to choose, 276
kryptonite, 270
multiple, 269, 279
pathological data set, 270
perfect, 289, 640
random, 271, 282
state-of-the-art, 276
universal, 270, 288, 640
hash map, *see* hash table
hash table
DELETE, 258, 267
INSERT, 258, 267, 268
LOOKUP, 258, 267, 268
OUTPUTUNSORTED, 287
advice, 273
applications, 259–263
as an array, 257, 264
bucket, 266
collision-resolution strategies, 276
for de-duplication, 260
for searching a huge state space, 262
for the 2-SUM problem, 260–262
hash function, *see* hash function
heuristic analysis, 274
in compilers, 259
in network routers, 259
in security applications, 270
iteration, 260, 287
load, 274
load vs. performance, 275
non-pathological data set, 270
operation running times, 259
performance of chaining, 267, 274
performance of open addressing, 269, 274–275
probe sequence, 267
raison d'être, 257
resizing to manage load, 275–276
scorecard, 259, 264, 275
space usage, 258
supported operations, 258
two-level, 640
vs. arrays, 263
vs. bloom filters, 277
vs. linked lists, 263
when to use, 259
with chaining, 266–267, 276
with double hashing, 268–269, 275
with linear probing, 268, 275, 276
with open addressing, 267–269, 276
head (of an edge), 143
heap (data structure)
DECREASEKEY, 225, 339
DELETE, 217, 236, 336
EXTRACTMAX, 217

©2022, Soundlikeyourself Publishing, LLC

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-0-999-28298-4 — Algorithms Illuminated
Tim Roughgarden
Index
More Information

EXTRACTMIN, 216, 232, 336
FINDMIN, 217
HEAPIFY, 217, 236, 638
INSERT, 216, 229, 336
applications, 218–222
as a tree, 226
as an array, 228
bubble/heapify/sift (up or down), 231, 234, 236
for an event manager, 220
for median maintenance, 220, 256
for sorting, 219
for speeding up Dijkstra's algorithm, 222–226
for speeding up Prim's algorithm, 336–339
heap property, 227
in Graham's algorithm, 474, 514, 648
in Huffman's algorithm, 319
in the LPT algorithm, 477
keys, 216
operation running times, 217, 336
parent-child formulas, 228
raison d'être, 216, 336
scorecard, 217
supported operations, 216–217
vs. search trees, 240–242
when to use, 217
heap (memory), 216
`HeapSort`, 219–220
Held, Michael, 524
Heule, Marijn, 543
heuristic algorithm, 293, 456, 631
dynamic programming, 515
greedy, 471–496, 600–604
local search, *see* local search
hill climbing, *see* local search
hints, xvi, 632–654
Hoare, Tony, 92
Hopcroft, John E., 5, 132
Huffman's algorithm, 313–316
Σ-tree, 312
average leaf depth, 312, 322
examples, 316–319
for ternary codes, 327
greedy criterion, 315
implemented with a heap, 319
implemented with two queues, 319, 327, 641
obtaining symbol frequencies, 309
proof of correctness, 320–325
pseudocode, 316
running time, 319
Huffman, David A., 313

i.e., 48
Impagliazzo, Russell, 588
in-place algorithm, 90
independence (in probability), 282, 625
independent set (of a graph), 366, 455
independent set problem, 558
is NP-hard, 558
reduces to clique, 575, 652
reduces to subset sum, 570–573
reduces to vertex cover, 575, 652
reduction from 3-SAT, 559–562
weighted, *see* weighted independent set
induction, *see* proofs, by induction
in greedy algorithms, 299
inductive hypothesis, 618
inductive step, 617
Indyk, Piotr, 589
influence, 492
is a weighted average of coverage functions, 494
is submodular, 515
influence maximization, 490–496
and $\mathcal{NP}$, 584
approximate correctness of greedy algorithm, 493–496
cascade model, 490
generalizes maximum coverage, 492, 514, 648
greedy algorithm, 493
intuition, 494
is NP-hard, 492, 555
problem definition, 492
running time of greedy algorithm, 493, 496
`InsertionSort`, 11, 21, 112
integer multiplication, 4–9, 71–73
grade-school algorithm, 4
Karatsuba's algorithm, 9
simple recursive algorithm, 7
integer programming, *see* mixed integer programming
interview questions, xv
intractable, *see* NP-hardness
invariant, 94
inversion, 46
left vs. right vs. split, 48
IQ points, 3

Jarník's algorithm, *see* Prim's algorithm
Jarník, Vojtěch, 331
Java, 240, 272, 327
job, *see* scheduling
Johnson's algorithm, 437
Johnson, David S., 557, 593

k-coloring, 541
k-SAT, 548
and the SETH, 588
Schöning's algorithm, 548–550
`Karatsuba`, 9
implementation, 26
recurrence, 72

©2022, Soundlikeyourself Publishing, LLC

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-0-999-28298-4 — Algorithms Illuminated
Tim Roughgarden
Index
More Information

running time, 76
Karatsuba multiplication, 6–9
in Python, 77
Karatsuba, Anatoly, 6
Karp reduction, 590, *see also* Levin reduction
Karp, Richard M., 524, 553
Karpman, Pierre, 543
Kempe, David, 492
KenKen, 452, 580
vs. Sudoku, 112
key, 216
key-value pair, 11
Kleinberg, Jon, 45, 192, 392, 492
knapsack problem, 213, 380–387
applications, 382
as a mixed integer program, 538, 540
correctness, 385
definition, 381, 455
dynamic programming algorithm, 384, 455, 469, 515, 649
example, 385–386
generalizations, 391
greedy algorithm, 512, 648
is NP-hard, 575, 652
is pseudopolynomial-time solvable, 455, 570
measuring input size, 455
optimal substructure, 382
reconstruction, 386–387
recurrence, 383
reduction from subset sum, 575, 652
running time, 385
subproblems, 384
two-dimensional, 539
Knuth Prize, 553
Knuth, Donald E., 39, 275, 410, 543, 593
`Kosaraju`, 187
Kosaraju's algorithm
correctness, 189
example, 188
from 30,000 feet, 183
implementation, 187, 197
pseudocode, 187
running time analysis, 189
why the reversed graph?, 184–186
Kosaraju, S. Rao, 183
Kowalik, Michał, 534
Kruskal's algorithm
achieves the minimum bottleneck property, 358
and clustering, 361
cycle-checking, 349, 351
example, 346
in reverse, 363
outputs a spanning tree, 357
proof of correctness, 357–358, 364
pseudocode (straightforward), 347
pseudocode (union-find-based), 351
reasons to care, 346
running time (straightforward), 348–349
running time (union-find-based), 349
stopping early, 348
vs. Prim's algorithm, 346
Kruskal, Joseph B., 346
Kumar, Ravi, 191

Ladner's theorem, 595
Lehman, Eric, 617
Leighton, F. Thomson, 617
Leiserson, Charles E., 252
lemma, 15
length
of a path, 198
of an edge, 198, 415
Lenstra, Jan Karel, 508
Levin reduction, 590–591
spreads NP-completeness, 592
transitivity of, 594, 653
Levin, Leonid, 553
Leyton-Brown, Kevin, 596
Lin, Shen, 593
Lin-Kernighan heuristic, 508, *see also* local search
linear programming, 539
linear signaling pathways (in a PPI network), 525
linear-time algorithm, 23
linearity of expectation, 274, 624
doesn't need independence, 108, 625
Linux kernel, 240
literal (in satisfiability), 542
little-o notation, 39
$\ln x$, 75
local search, 497–510
2-OPT algorithm, *see* 2-OPT algorithm
and gradient descent, 504
as a walk in a graph, 504
as no-downside postprocessing, 505, 631
avoiding bad local optima, 508
choosing a neighborhood, 506
choosing an improving local move, 507
feasible solutions, 505
for maximum k-cut, 517, 649
for satisfiability, 608
for the TSP, 497–501, 506
generic pseudocode, 504
global optima, 505
history-dependent neighborhoods, 508
initialization, 507
is interruptible, 507
local moves, 505
local optima, 505
meta-graph, 502–504
non-improving local moves, 508
objective function, 505

©2022, Soundlikeyourself Publishing, LLC

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-0-999-28298-4 — Algorithms Illuminated
Tim Roughgarden
Index
More Information

overview of paradigm, 504–505
population of solutions, 508
running time, 507
solution quality, 507
vs. MIP and SAT solvers, 509
when to use, 509, 631
logarithms, 15, 74
Lokshtanov, Daniel, 534
longest common subsequence, 413, 459
reduces to sequence alignment, 645
longest common substring, 413
longest processing time first, *see* LPT algorithm
lower-order terms, 28
LPT algorithm, 477
approximate correctness, 478–479, 514, 648
bad example, 477, 480, 514, 648
intuition, 478
running time, 477

machine learning, 504
supervised learning, 358
unsupervised learning, *see* clustering
Maghoul, Farzin, 191
magic boxes, 535–537, *see also* MIP solvers, SAT solvers
and NP-hardness, 537
when to use, 631
makespan minimization, 471–481
as a mixed integer program, 547, 651
Graham's algorithm, *see* Graham's algorithm
in practice, 472, 478
is NP-hard, 472, 576, 653
LPT algorithm, *see* LPT algorithm
machine load, 471
objective function, 471
problem definition, 472
reduction from subset sum, 576, 653
with small jobs, 511, 647
mangosteen, 112
mantra, 5
Markov, Yarik, 543
Marx, Dániel, 534
master method
a, b, and d, 73, 79
applied to `RecIntMult`, 76
applied to `Karatsuba`, 76
applied to `MergeSort`, 75
applied to `Strassen`, 77, 78
applied to binary search, 75, 78
big-theta vs. big-O, 74
does not apply, 103, 106, 136
formal statement, 74
meaning of the three cases, 82–84
more general versions, 74
proof, 80–86
master theorem, *see* master method
mathematical background, xv, 616–628
matrix multiplication
definition, 53
exponent, 79
iterative algorithm, 54
simple recursive algorithm, 56
Strassen's algorithm, 56–58, 452
the 2×2 case, 54
matrix-vector multiplication, 69
max-heap, *see* heap (data structure), EXTRACT-MAX
maximum k-cut problem, 517, 649
maximum coverage, 481–490
applications, 483
approximate correctness of greedy algorithm, 486–489
as a mixed integer program, 547, 651
as team-hiring, 481
bad example, 482, 484–486, 490, 514, 648
greedy algorithm, 484
hardness of approximation, 487
intuition, 487
is NP-hard, 576, 652
problem definition, 482
reduction from set cover, 576, 652
running time of greedy algorithm, 484
MBP, *see* minimum spanning tree, minimum bottleneck property
median (of an array), 26, 99, 121, 220
approximate, 103, 127, 621
vs. mean, 121
weighted, 139
median-of-medians, *see* `DSelect`
memoization (in dynamic programming), 373
`Merge`, 13–14
for counting inversions, 49
running time, 14–15
`MergeSort`, 10–20
analysis, 16–19
as a divide-and-conquer algorithm, 45
does not run in place, 90
implementation, 70
is comparison-based, 112
motivation, 10
pseudocode, 12
recurrence, 73
running time, 15, 75
metric TSP, *see* traveling salesman problem, metric instances
Metropolis algorithm, 508, *see also* local search
Meyer, Albert R., 593, 617
Milgrom, Paul, 596
Millennium Problems, 586
min-heap, *see* heap (data structure)
minimum bottleneck property, *see* minimum spanning tree, minimum bottleneck property

©2022, Soundlikeyourself Publishing, LLC

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-0-999-28298-4 — Algorithms Illuminated
Tim Roughgarden
Index
More Information

minimum spanning tree
  Cut Property, 340, 364, 642
  Cycle Property, 364, 642
  exchange argument, 345
  history, 331
  in directed graphs, 328
  in disconnected graphs, 330
  in linear time?, 643
  Kruskal's algorithm, *see* Kruskal's algorithm
  minimum bottleneck property, 340, 341, 344, 363
  MST heuristic (for the TSP), 516
  Prim's algorithm, *see* Prim's algorithm
  reductions to, 363
  uniqueness, 364
  with distinct edge costs, 340, 344
  with non-distinct edge costs, 342, 357, 363, 642
  with parallel edges, 330
minimum-bottleneck spanning tree, 364, 643
minimum-cost k-path problem, *see also* color coding
  exhaustive search, 527, 535
  is NP-hard, 526, 555
  problem definition, 526
minimum-cost panchromatic path problem, *see* color coding
MiniSAT, *see* SAT solvers
MIP, *see* mixed integer programming
MIP solvers, 537–540
  and modeling languages, 540
  and nonlinearities, 539
  are interruptible, 540
  branch and bound, 536
  example input file, 538
  for the TSP, 548, 550
  in the FCC Incentive Auction, 600, 602
  starting points, 540
  vs. local search, 509
  when to use, 537, 631
mixed integer programming, 537
  and linearity, 539
  constraints, 538
  decision variables, 538
  for knapsack, 538, 540
  for makespan minimization, 547, 651
  for maximum coverage, 547, 651
  for satisfiability, 548, 652
  for the TSP, 547, 550, 651
  for weighted independent set, 547, 651
  is NP-hard, 555
  multiple formulations, 540
  objective function, 538
  problem definition, 539
  solvers, *see* MIP solvers
mod (operator), 272
mode (of an array), 26
modulo, 272
Moore's law, 3, 22, 449, 465
MP3, 306, 309
MST, *see* minimum spanning tree
MWIS, *see* weighted independent set

$n \log n$ vs. n^2, 16, 22
Nash, John F., Jr., 586
nearest neighbors (in computational geometry), 70
Needleman, Saul B., 393
Needleman-Wunsch (NW) score, *see* sequence alignment
Nemhauser, George L., 484
network
  movie, 153
  neural, 504
  physical, 153
  road, 143
  social, 143
neural networks, 504
Nielsen, Morten N., 291
Nilsson, Nils J., 212
Nobel Prize, *see* Turing Award
node, *see* vertex (of a graph)
nondeterminism, *see* $\mathcal{NP}$
NP (acronym), 465, 580
$\mathcal{NP}$ (complexity class)
  and $\mathcal{FNP}$, 579
  and efficient recognition of solutions, 580
  and nondeterministic Turing machines, 580
  as problems solvable by naive exhaustive search, 579, 581
  does not contain the halting problem, 579
  feasible solution, 579
  formal definition, 580
  search vs. decision, 579
NP-completeness, 590–593
  formal definition, 591
  meaning, 590
  three-step recipe, 592, 594, 653
  vs. NP-hardness, 466, 591
  what's in a name?, 593
  with decision problems, 591
  with Karp reductions, 591
NP-hardness
  acceptable inaccuracies, 466
  and magic boxes, 537
  and Moore's law, 449, 465
  and reductions, 460, 551
  and subexponential-time solvability, 595, 653
  applies to randomized and quantum algorithms, 452
  as amassing evidence of intractability, 450, 577–579
  as relative intractability, 450

©2022, Soundlikeyourself Publishing, LLC

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-0-999-28298-4 — Algorithms Illuminated
Tim Roughgarden
Index
More Information

definition (formal), 582
definition (provisional), 452
expertise levels, 447–448
forces compromise, 454
in practice, 465
in fifteen words, 448
in other disciplines, 551
is not a death sentence, 454, 465
is ubiquitous, 454
key takeaways, 457
level-1 expertise, 448–457, 464–466
level-2 expertise, 453–457, 471–550, 596–616
level-3 expertise, 457–464, 551–576
level-4 expertise, 577–595
main idea, 451
of 19 problems, 554–557
of 3-SAT, *see* Cook-Levin theorem
oversimplified dichotomy, 448, 453, 466, 595, 654
proofs, 556
rookie mistakes, 464–466
strong vs. weak, 570
subtleties, 453
two-step recipe, 461, 552, 594, 631, 653
vs. NP-completeness, 466, 554, 591
with Levin reductions, 582
null pointer, 241

$O(f(n))$, *see* big-O notation
$o(f(n))$, *see* little-o notation
O'Donnell, Ryan, 577
objective function, 293
$\Omega(f(n))$, *see* big-omega notation
optimal binary search trees, 400–410
correctness, 409
dynamic programming algorithm, 408
Knuth's optimization, 410
optimal substructure, 403–406
problem definition, 403
reconstruction, 409
recurrence, 406
running time, 409
search frequencies, 402
subproblems, 408
vs. balanced binary search trees, 402
vs. optimal prefix-free codes, 403
weighted search time, 403
with unsuccessful searches, 403
optimization problem, 579
reduces to search version, 570, 575, 580, 652
order statistic, 107, 120

$\mathcal{P}$ (complexity class), 584
P $\neq$ NP conjecture, 451–452
as a law of nature, 466
current status, 585–586
formal definition, 585
informal version, 451
is hard to prove, 452, 586
meaning, 585
reasons to believe, 586
refutation consequences, 586
vs. the ETH, 453, 587
panchromatic path, 528
Papadimitriou, Christos, 73
`Partition`, 96
proof of correctness, 97
runs in place, 98
partition problem, *see* subset sum problem
path (of a graph), 329
k-, 526
bottleneck, 341
cycle-free, 329
panchromatic, 528
path graph, 368
pathological data set, 270
Paturi, Ramamohan, 588
paying the piper, 224, 251, 339
pep talk, 33, 366
Perelman, Grigori, 586
PET, 593
Pigeonhole Principle, 265, 270, 418
Pilipczuk, Marcin, 534
Pilipczuk, Michał, 534
pivot element, 91
planning (as graph search), 154
pointer, 147
polynomial vs. exponential time, 449
polynomial-time algorithm, 448
polynomial-time solvability, 450
and reductions, 460
Pratt, Vaughan, 132
prefix-free code, *see* code, prefix-free
Prim's algorithm
achieves the minimum bottleneck property, 341
example, 331
greedy criterion, 333
outputs a spanning tree, 344
proof of correctness, 340–345, 364
pseudocode, 333
pseudocode (heap-based), 338
resembles Dijkstra's shortest-path algorithm, 331
running time (heap-based), 336, 339
running time (straightforward), 335
starting vertex, 334
straightforward implementation, 335
vs. Dijkstra's shortest-path algorithm, 334, 340
vs. Kruskal's algorithm, 346
Prim, Robert C., 331

©2022, Soundlikeyourself Publishing, LLC

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE

Cambridge University Press & Assessment
978-0-999-28298-4 — Algorithms Illuminated
Tim Roughgarden
Index
More Information

prime number, 102
primitive operation, 4, 14, 19, 21
principle of parsimony, 215, 630
priority queue, *see* heap (data structure)
probability, 274, 490, 532, 620
  independence, 282
  of an event, 621
problem
  NP-hard (formal definition), 582
  NP-hard (provisional definition), 452
  decision, 579
  easy (oversimplified), 448
  easy vs. polynomial-time solvable, 450
  hard (oversimplified), 448
  neither polynomial-time solvable nor NP-hard, 453, 595
  optimization, 579
  polynomial-time solvable, 449
  search, 579
  undecidable, 578
  universal, 592
problems vs. solutions, 3
programming, xv, 9
programming problems, xvi
proofs, xv
  by contradiction, 36, 158, 300
  by induction, 93, 137, 617–618
  of correctness, 93, 298
  on reading, 206
proposition, 15
protein-protein interaction (PPI) networks, 525
pseudocode, 9, 13
pseudopolynomial time, 570
Pyrrhic victory, 135
Pythagorean theorem, 67

Q.E.D. (q.e.d.), 19
quantum algorithm, 452
queue (data structure), 160, 214, 319
`QuickSort`
  best-case scenario, 99
  handling ties, 91
  high-level description, 92
  history, 92
  implementation, 118
  is comparison-based, 112
  is not stable, 113
  median-of-three, 118
  partitioning around a pivot, 91, 93–98
  pivot element, 91
  proof of correctness, 93
  pseudocode, 98
  random shuffle, 102
  randomized, 101
  running time, 102
  running time (intuition), 103–104
  running time (proof), 104–111
  runs in place, 90
  worst-case scenario, 99
quizzes, xvi

Rackoff, Charles, 291
`RadixSort`, 113
Raghavan, Prabhakar, 191
Rajagopalan, Sridhar, 191
random variable, 622
  geometric, 128
  independent, 625
  indicator, 627
randomized algorithms, 102, 129, 628, 631
  and NP-hardness, 452
  for 3-SAT, 548
  for color coding, 531–533
Raphael, Bertram, 212
Rassias, Michael Th., 586
rate of growth, *see* asymptotic analysis
`RecIntMult`, 7
  recurrence, 72
  running time, 76
`RecMatMult`, 56
recommendation system, 47
recurrence, 71, 371
  standard, 73
recursion, 7
recursion tree, 16, 80
reduction, 121, 201, 204, 458, 551, 630
  Cook, 578
  examples, 458
  gone awry, 562
  in the wrong direction, 465, 554
  Karp, 590
  Levin, 590–591
  many-to-one, 590
  mapping, 590
  polynomial-time Turing, 578
  preprocessor and postprocessor, 558, 591
  preserves polynomial running time, 459, 464
  simplest-imaginable, 557–558
  spreads intractability, 460, 552
  spreads tractability, 460, 536, 552
  to a magic box, 536
  transitivity of, 594, 653
  with exponential blow-up, 590, 654
reference, *see* pointer
reverse auction, *see* FCC Incentive Auction
Rivest, Ronald L., 132, 252
rookie mistakes, 84, 464–466
`RSelect`
  best-case scenario, 124
  expected running time, 125
  implementation, 140
  pseudocode, 122

©2022, Soundlikeyourself Publishing, LLC

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-0-999-28298-4 — Algorithms Illuminated
Tim Roughgarden
Index
More Information

running time analysis, 126–129
runs in place, 123
worst-case scenario, 124
RSP (rate of subproblem proliferation), *see* master method, meaning of the three cases
running time, 14, 19, 28
RWS (rate of work shrinkage), *see* master method, meaning of the three cases

sample space (in probability), 620
SAT, *see* satisfiability
SAT solvers, 540–544
and graph coloring, 616
and local search, 608
and the FCC Incentive Auction, 604–609
applications, 542
are only semi-reliable, 553
conflict-driven clause learning, 536
example input file, 543
portfolio, 608
starting points, 544
vs. local search, 509
when to use, 541, 631
satisfiability
k-SAT, *see* k-SAT
2-SAT (is linear-time solvable), 197, 548
3-SAT, *see* 3-SAT
and graph coloring, 542–543, 604
and the SETH, 588
applications, 542
as a mixed integer program, 548, 652
constraints, 542
decision variables, 542
disjunction (of literals), 542
is NP-hard, 555
literal, 542
modulo theories, 544
problem definition, 542
truth assignment, 542
Saurabh, Saket, 534
SCC, *see* strongly connected components
scheduling, 293, 471, *see also* makespan minimization
`GreedyDiff`, 297
`GreedyRatio`, 297
completion time, 293
correctness of `GreedyRatio`, 299–302
exchange argument, 300
greedy algorithms, 295–298
running time, 298
sum of weighted completion times, 294
Schrijver, Alexander, 418
Schöning's algorithm (for k-SAT), 548–550
and the ETH, 587
and the SETH, 588
Schöning, Uwe, 550
scorecards, 150, 217, 238, 240, 259, 264, 275, 278, 350
search problem, 579
reduces to decision version, 562, 575, 652
search tree
DELETE, 239, 247, 251
INSERT, 239, 246, 251
MAX, 237, 244
MIN, 237, 244
OUTPUTSORTED, 237, 246
PREDECESSOR, 237, 245
RANK, 238, 249, 255
SEARCH, 237, 243
SELECT, 237, 249–251
SUCCESSOR, 237, 245
2-3, 251
applications, 240
augmented, 249, 251, 255
AVL, 251
B, 251
balanced, 240, 251–253
height, 243
in-order traversal, 246
operation running times, 240
optimal, *see* optimal binary search trees
pointers, 241
raison d'être, 239
red-black, 251, 252
rotation, 252–253
scorecard, 240
search tree property, 241
splay, 251
supported operations, 239
vs. heaps, 240–242
vs. sorted arrays, 237, 240
when to use, 240
with duplicate keys, 255
Sedgewick, Robert, 91, 252
Segal, Ilya, 596
selection
`DSelect`, *see* `DSelect`
`RSelect`, *see* `RSelect`
problem definition, 120
reduces to sorting, 121
selection bias, 442
`SelectionSort`, 11, 112, 219
separate chaining, *see* hash table, with chaining
sequence alignment, 392–399
alignment, 393
and the SETH, 589
applications, 392
correctness, 398
dynamic programming algorithm (`NW`), 397
gap, 393
Needleman-Wunsch (NW) score, 393
optimal substructure, 394

©2022, Soundlikeyourself Publishing, LLC

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-0-999-28298-4 — Algorithms Illuminated
Tim Roughgarden
Index
More Information

penalties, 393
problem definition, 393
reconstruction, 399
recurrence, 396
reduction from longest common subsequence, 645
running time, 398
subproblems, 397
variations, 413
set cover problem, 511
greedy algorithm, 511, 647
is NP-hard, 575, 652
reduces to maximum coverage, 576, 652
reduction from vertex cover, 575, 652
SETH, *see* Strong Exponential Time Hypothesis
Shamir, Adi, 132
Sharir, Micha, 183
Shimbel, Alfonso, 418
shortest paths
all-pairs, 429, 459
all-pairs (dense graphs), 437
all-pairs (sparse graphs), 430, 437
and Bacon numbers, 154
and Internet routing, 427
and negative cycles, 462
and transitive closure, 429
Bellman-Ford algorithm, *see* Bellman-Ford algorithm, 462
bottleneck, 212, 236
cycle-free, *see* cycle-free shortest paths
Dijkstra's algorithm, *see* Dijkstra's shortest-path algorithm
distance, 163, 198, 415
Floyd-Warshall algorithm, *see* Floyd-Warshall algorithm
history, 418
Johnson's algorithm, 437
nonnegative edge lengths, 200, *see also* Dijkstra's shortest-path algorithm
problem definition (all-pairs), 429
problem definition (single-source), 163, 198, 417
reduction from all-pairs to single-source, 429, 430
single-source, 198, 415, 459, 461
via breadth-first search, 164–165, 201
with negative cycles, 417
with negative edge lengths, 204, 416
with no negative cycles, 418
with parallel edges, 416
with unit edge lengths, 163, 201
SIGACT, 39
simulated annealing, 508, *see also* local search
single-source shortest path problem, *see* shortest paths, single-source
six degrees of separation, 192
small world property, 192
social network, 490
solutions, xvi, 632–654
solver, 536, *see also* magic box
sorted array
scorecard, 238
supported operations, 237
unsupported operations, 239
vs. search trees, 240
sorting, 458
`HeapSort`, *see* `HeapSort`
`MergeSort`, *see* `MergeSort`
`MergeSort` vs. `QuickSort`, 90
`QuickSort`, *see* `QuickSort`
applications, 11, 26
associated data, 11
by key, 11
comparison-based, 112
in linear time, 320
in place, 90
in Unix, 112
lower bound, 112, 114–115
non-comparison-based, 113
problem definition, 11, 218
randomized, 102, 114
simple algorithms, 11–12
stable, 113
with duplicates, 11
with Hungarian folk dancers, 96
spanning tree (of a graph), 329
component fusion, 342
cycle creation, 342
minimum, *see* minimum spanning tree
number of, 330
number of edges, 343
type-C vs. type-F edge addition, 342
spectrum auction, *see* FCC Incentive Auction
stack (data structure), 172, 214
pop, 172
push, 172
stack (memory), 173
starred sections, xiv, 59
Stata, Raymie, 191
Steiglitz, Kenneth, 593
Stein, Clifford, 252
Stevens, Marc, 543
Stirling's approximation, 519, 532
`Strassen`, 56–58, 452
running time, 77, 78
Strassen, Volker, 58
strong NP-hardness, 570
Strong Exponential Time Hypothesis
and graph diameter, 595, 654
and Schöning's algorithm, 550, 588
and sequence alignment, 589
definition, 588

©2022, Soundlikeyourself Publishing, LLC

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-0-999-28298-4 — Algorithms Illuminated
Tim Roughgarden
Index
More Information

vs. the ETH, 588
strongly connected components
and the 2-SAT problem, 197
definition, 181
giant, 191
in a reversed graph, 186, 189
linear-time computation, *see* Kosaraju's algorithm
meta-graph of, 182
sink, 184
source, 185
topological ordering of, 182, 185
via depth-first search, 181, 183
submodular function maximization, 514–515, 649
greedy algorithm, 514
subsequence, 413
subset sum problem
is NP-hard, 570
is pseudopolynomial-time solvable, 570
partition special case, 576
problem definition, 569
reduces to knapsack, 575, 652
reduces to makespan minimization, 576, 653
reduction from independent set, 570–573
substring, 413
Sudoku, 154, 212, 452, 580
vs. KenKen, 112
superteam, 132

tabu search, 508, *see also* local search
tail (of an edge), 143
Tardos, Éva, 45, 392, 492
Tarjan, Robert E., 132, 181, 638
task scheduling, 143, 174, 175
team-hiring, *see* maximum coverage
test cases, xvi
Tetris, 413
theorem, 15
$\Theta(f(n))$, *see* big-theta notation
theta notation, *see* big-theta notation
three-step recipe, *see* NP-completeness, three-step recipe
Tomkins, Andrew, 191
topological ordering
definition, 174
existence in directed acyclic graphs, 176–177
non-existence, 175
topological sorting, 174–180
example, 178
linear-time computation, 179
problem definition, 177
pseudocode, 178
`TopoSort`, 178
correctness, 179
in non-acyclic graphs, 181, 184
run backward, 187
running time analysis, 179
tour (of a graph), 443
transitive closure (of a binary relation), 429
traveling salesman problem
2-OPT algorithm, *see* 2-OPT algorithm
2-change, 499
3-OPT algorithm, 506
3-change, 506
applications, 445
as a mixed integer program, 547, 550, 651
Concorde solver, 548
conjectured intractability, 446
dynamic programming, *see* Bellman-Held-Karp algorithm (for the TSP)
exhaustive search, 444, 446, 470, 579
history, 445
is NP-hard, 497, 568
metric instances, 516–517, 576, 649
MST heuristic, 516, 649
nearest neighbor algorithm, 497, 501, 518
number of tours, 443, 446
on non-complete graphs, 443
optimal substructure, 521–522, 524–525
path version, 445, 469, 555
problem definition, 443
reduction from undirected Hamiltonian path, 568–569
search version, 575
tree instances, 470, 647
tree, 146
binary, 226
chain, 243
complete, 226
depth (of a node), 311
forest, 315
height, 240, 243
optimal binary search, *see* optimal binary search trees
root, 226
search, *see* search tree
TSP, *see* traveling salesman problem
tug-of-war, 82
Turing Award, 92, 132, 553
Turing machine, 578
nondeterministic, 580
Turing reduction, 578
Turing, Alan M., 578
two-step recipe, *see* NP-hardness, two-step recipe

`UCC`, 168
correctness, 169
running time analysis, 169
Ullman, Jeffrey D., 5
uniform distribution, 620
union-find
FIND, 350, 353

©2022, Soundlikeyourself Publishing, LLC

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-0-999-28298-4 — Algorithms Illuminated
Tim Roughgarden
Index
More Information

INITIALIZE, 350, 353
UNION, 350, 354, 355
for speeding up Kruskal's algorithm, 351–352
inverse Ackermann function, 350
operation running times, 350, 356
parent graph, 352
path compression, 350
quick-and-dirty implementation, 352–356
raison d'être, 349
scorecard, 350
state-of-the-art implementations, 350
supported operations, 350
union-by-rank, 350, 355
union-by-size, 355
Unix, 413
upshot, xiv

van Maaren, Hans, 543
Vazirani, Umesh, 73
vertex (of a graph), 142
degree, 151
in-degree, 421
out-degree, 421
reachable, 155
sink, 176
source, 176, 198
starting, 198
vertex cover problem, 513
greedy algorithm, 513, 648
is NP-hard, 575, 652
reduces to set cover, 575, 652
reduction from independent set, 575, 652
videos, xvi, 252, 275, 289, 350, 355, 427, 428, 437, 640, 643, 649
von Neumann, John, 10, 586

Wallach, Dan S., 270
Walsh, Toby, 543
Warshall, Stephen, 430
Wayne, Kevin, 91, 252
weak NP-hardness, 570
Web graph, 143, 190–192
as a sparse graph, 149
bow tie, 191
connectivity, 192
giant component, 191
size, 149, 191, 259
weighted independent set
as a mixed integer program, 547, 651
greedy algorithm, 600–602, 616, 654
in acyclic graphs, 455
in general graphs, 390
in path graphs, *see* weighted independent set (in path graphs)
in the FCC Incentive Auction, 599
problem definition, 367, 454
weighted independent set (in path graphs), 368
correctness, 375
dynamic programming algorithm, 374
failure of divide-and-conquer algorithms, 369
failure of greedy algorithms, 368, 370
optimal substructure, 370–371
reconstruction, 376–377
recurrence, 371
recursive algorithm, 372
running time, 374
subproblems, 373
Wernicke, Sebastian, 535
whack-a-mole, 229
why bother?, xiv, 2
Wiener, Janet, 191
Williams, H. Paul, 540
Williams, Virginia Vassilevska, 589
WIS, *see* weighted independent set
work, *see* running time
World Wide Web, *see* Web graph
worst-case analysis, 20
Wunsch, Christian D., 393

yottabyte, 259
YouTube, xvi
Yuster, Raphael, 527

Zichner, Thomas, 535
Zwick, Uri, 527

©2022, Soundlikeyourself Publishing, LLC

© in this web service Cambridge University Press & Assessment www.cambridge.org