Bellman–Ford algorithm, 213
big O notation, 407
correspondent time (O(1)), 408–409
exponential time, 86
linear time (O(n)), 20, 408
logarithmic time (O(log n)), 242
O(n log n), 299, 400–401
quadratic time (O(n^2)), 10, 23, 410
ingredient, easy feasibility, 240–241
ingredient, feasibility split, 241
time, 241–242
on sorted array, 242–243, 379–381
binary search tree, property, 301
binary tree, 39, 41. See also tree
building, 43–45
full, 41
left child, 40
node struct, 42
right child, 40
See also graph
adjacency list in graph, 190
modeling as graph, 189–190
reading line, 188–189
bottom-up solution, 97
breath-first search (BFS), 159
0-1, 186
time, 172–173
time optimization, 168
bucket in hash table, 17
Building Treaps problem, 300–317
binary search tree, property, 301
comparison function for
sorting, 305
range maximum query, 307–308
segment tree, 308
height, 309
maximum index, 312–313

A
AC (accepted) status, xxx
add_position function, 161–162, 417
adjacency list, 190, 203
adjacency matrix, 214
algorithm, xxiv. See also binary
search; breadth-first search;
Dijkstra’s algorithm; dynamic
programming; randomized
algorithm; recursion
Bellman–Ford, 213
correspondent time (O(1)), 408–409
deterministic, 389
divide and conquer, 394
equal runtime, 400
exponential-time, 86
Floyd–Warshall, 211
greedy, 100, 244, 247
linear time (O(n)), 20, 408
logarithmic-time (O(log n)), 242
memoization, 88–89, 96
O(n log n), 299, 400–401
quadratic-time (O(n^2)), 10, 23, 410
Algorithm Design (Kleinberg and
Tardos), 123
Algorithms Illuminated (Part 1): The Basics
(Roughgarden), 275
Algorithms Illuminated (Part 2): Graph
Algorithms and Data Structures
(Roughgarden), 36, 195
Algorithms Live videos (Fontaine), 329
allocating memory, xxvii
all-pairs shortest-paths, 211
augmenting in data structure, 356
avalanche effect in hash table, 19

B
base case in recursion, 55
base-2 logarithm, 241–242
Building Treaps problem (continued)
  segment tree (continued)
    querying, 313–316
    runtime, 315
    segments of, 311–312
  treap, 300–301
Burger Fervor problem, 78–94.
  See also dynamic programming
  memoization, 88–89
  optimization problem, 80
  solutions
    feasible, 80
    infeasible, 80
    optimal, 80
    reconstructing, 414–416
    recovering, 414–416

C
  C (programming language), xxvi
  call stack, xxvi
  Canadian Computing Competition (CCC), xxix
  Canadian Computing Olympiad (CCO), xxix
  Caps and Bottles problem, 390–398.
    See also randomized algorithm
    divide and conquer (D&C), 394
    expected runtime, 402
    in place, 424
    randomization, 397–398
    subtask, 391
  Cave Doors problem, 267–274.
    See also binary search
    linear search, 272
    subtask, 268
  chaining in hash table, 19
  children in tree, 40
  Codeforces judge, xxix
  collision in hash table, 18
  Compared to What: An Introduction to the Analysis of Algorithms (Rawlins), 402
  comparison function for sorting, 10–11, 74, 254, 305
  Competitive Programming 4 (Halim and Halim), 229
  complete
    in binary tree, 282
  D
  data structure, xxiv. See also binary tree; graph; hash table; heap; segment tree; union-find
  choosing, 300
  depth-first search (DFS), 195
  Descendant Distance problem, 66–76.
    See also recursion
    comparison function for sorting, 74
  deterministic algorithm, 389
  Dijkstra, Edsger W., 203
  Dijkstra's algorithm, 203, 210
    breadth-first search compared to, 210
    in heap, 418
    runtime, 210, 420–421
    shortest paths, number of, 224
  directed in graph, 170
  disconnected in graph, 171–172
  divide and conquer (D&C), 394
  DMOJ judge, xxix
    See also union-find
  DWITE contest, xxx
  dynamic programming, 92, 95–97
    memoization, compared to, 97
    optimal substructure, 95
    solutions
      backward, 134
      bottom-up, 97
      forward, 134
connected in graph, 171
constant-time algorithm (O(1)), 408–409
C Programming: A Modern Approach (King), xxv
Croatian Open Competition in Informatics (COCI), xxix
cycle in graph, 170
<table>
<thead>
<tr>
<th>Index</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>433</td>
</tr>
<tr>
<td>reconstructing</td>
<td>414–416</td>
</tr>
<tr>
<td>recovering</td>
<td>414–416</td>
</tr>
<tr>
<td>top-down</td>
<td>97</td>
</tr>
<tr>
<td>space optimization</td>
<td>122</td>
</tr>
<tr>
<td>subproblems</td>
<td></td>
</tr>
<tr>
<td>adding</td>
<td>145</td>
</tr>
<tr>
<td>“exactly,”</td>
<td>95–96, 114, 128, 140</td>
</tr>
<tr>
<td>order of</td>
<td>119, 137</td>
</tr>
<tr>
<td>overlapping</td>
<td>96</td>
</tr>
<tr>
<td>parameters</td>
<td>113–114, 128, 140</td>
</tr>
</tbody>
</table>

**E**

East Central North America (ECNA) contest, xxix–xxx

edges
- in graph, 169
- in tree, 40

equivalence relation in union-find, 353

expected runtime, 400

exponential-time algorithm, 86

extracting
- from max-heap, 287–289
- from min-heap, 294

extract-max, 280

extract-min, 280

**F**

feasible solution, 80, 237, 246, 259

Feeding Ants problem, 231–240
- binary search, 240
- feasible solution, 237

find function, 345, 351

find operation in union-find, 341–342

first-in, first-out (FIFO) in queue, 50

Floyd–Warshall algorithm, 211

Fontaine, Matt, 329

Food Lines problem, xxi–xxxiv
- input redirection, xxxiv

free function, xxvii

- See also union-find

full in binary tree, 41

functions
- add_position, 161–162, 417
- comparison, 10–11, 74, 254, 305
- extract_max, 280
- extract_min, 280
- find, 345, 351

free, xxvii

helper, xxiii–xxxiii, 5, 69, 106

main, xxix, 9, 258

malloc, xxvii

oaat, 18–19, 24, 36

qsort, 10–11, 74

rand, 378–379

solve, xxxiii, 85, 101

srand, 387

strcmp, 75

**G**

Grandma Planner problem, 213–228.
- See also Dijkstra’s algorithm

adjacency matrix in graph, 214

complete graph, 215

modeling as graph, 218–219

shortest paths, number of, 224

state in graph, 217

graph, 169. See also breadth-first search;
- Dijkstra’s algorithm

adjacency list, 190, 203

adjacency matrix, 214

all-pairs shortest-paths, 211

Bellman–Ford algorithm, 213

complete, 215

cycle, 170

directed, 170

disconnected, 171–172

dynamic programming, compared to, 173

dge, 169

Floyd–Warshall algorithm, 211


negative-weight edge, 211
	node, 169

reversed, 208

shortest paths, 172

- number of, 224

single-source shortest-paths, 210

- counting edges, 172

state, 182, 217

undirected, 170

weighted, 182, 199

greedy algorithm, 100, 244, 247
Halim, Felix, xxviii, 229
Halim, Steven, xxviii, 229

   See also recursion
      base case in recursion, 55
      binary tree, 39, 41
         building, 43–45
         full, 41
         left child, 40
         node struct, 42
         right child, 40
      first-in, first-out in queue, 50
      queue, 50
      reading integers from string, 62–63
      recursive, 52
         call, 54
      stack, 47
         implementation, 47–49
         last-in, first-out, 47
         pop, 47
         push, 47
         top, 47
      tree, 40
         binary, 41
         children, 40
         descendant, 40
         edge, 40
         height, 40
         leaf, 40
         node, 40
         parent, 40
         root, 40
         siblings, 40
         subtrees, 40
         vertex, 40
      heap, 298
         as array, 291
         complete, 282
         in Dijkstra’s algorithm, 418
         height, 290
         max-heap, 282–283
            extracting from, 287–289
            inserting into, 283–287
            order, 283
         min-heap, 293
            extracting from, 294
            inserting into, 294
            order, 294
            runtime, 290
      heapsort, 299
      height
         of heap, 290
         of segment tree, 309
         in tree, 40
      helper function, xxxii–xxxiii, 5, 69, 106
      Hockey Rivalry problem, 108–123.
         See also dynamic programming
            maximization problem, 115
            memoization, 118
            space optimization in dynamic programming
               subproblems
               “exactly,” 114
               order of, 119–120
               parameters, 113–114

I
   identical integers, checking for, 4
   include files, xxvii
   incremental hash function in hash table, 31
   infeasible solution, 80
   input of problem description, xxx
   input redirection, xxxiv
   inserting
      into max-heap, 283–287
      into min-heap, 294
International Olympiad in Informatics (IOI), xxix
invariant, 252, 259
inverse Ackermann function in union-find, 352

J
Jenkins, Bob, 36
judges, xxviii
Codeforces judge, xxix
DMOJ judge, xxix
POJ judge, xxix
programming judge, xxviii
SPOJ judge, xxix
UVa judge, xxix

The Jumper problem, 125–137.
  See also dynamic programming
  memoization, 130
  solutions, 134
subproblems
  “exactly,” 128
  order of, 137
  parameters, 128

K
King, K. N., xxv
Kleinberg, Jon, 123
Knight Chase problem, 151–169.
  See also graph
  breadth-first search, 159
  parity of an integer, 167
  time optimization, 168

L
last-in, first-out (LIFO) in stack, 47
leaf in tree, 40
left child in binary tree, 40
linear search, 27, 69, 272
linear-time algorithm \(O(n)\), 20, 408
linked list
  adding to, 15
  node struct in, 14
  See also binary search
  feasible solution, 259
  invariant, 259
  median, calculating, 256
  range sum query, 262
    one dimension, 262–263
    two dimensions, 263–266
  logarithm, 241–242
  logarithmic-time algorithm \(O(\log n)\), 242
Login Mayhem problem, 20–29.
  See also hash table
  linear search, 27
  oaat hash function, 24
  longest common prefix, 32
    of strings, 33
  longest common suffix, 32
    of strings, 33–34

M
main function, xxxiii, 9, 258
malloc function, xxvii
max-heap, 282–283
maximization problem, 102
maximum index in segment tree, 312–313
maximum sum of two elements in segment tree, 319–323
median, calculating, 256
memoization, 88–89, 96.
  See also dynamic programming
memory allocation, xxviii
Methods to Solve page (Halim and Halim), xxviii
Mice Maze problem, 198–209.
  See also Dijkstra’s algorithm
  adjacency list in graph, 203
  modeling as graph, 199
  reversed in graph, 208
  weighted in graph, 199
min-heap, 293
minimization problem, 102
modeling as graph, 173, 180–182, 189–190, 199, 218–219, 333
Moneygrubbers problem, 98–108.
  See also dynamic programming
  greedy algorithm, 100
  maximization problem, 102
  memoization, 108
  minimization problem, 102
  optimal substructure, 100
  reading integers from line, 106
N
National Olympiad in Informatics in Province (NOIP), xxix
node
   in graph, 169
   in tree, 40
node struct
   in binary tree, 42
   in linked list, 14

O
O(1) (constant time), 408–409
oaat (one-at-a-time) hash function, 36
   in hash table, 18–19, 24
O(n^2) (quadratic time), 10, 23, 410
O(log n) (logarithmic time), 242
O(n) (linear time), 20, 408
O(n log n), 299, 400–401
open addressing in hash table, 19–20
optimal solution, 80
optimization problem, 80
output of problem description, xxxi

P
parent in tree, 40
parity of an integer, 167
path compression in union-find, 351, 422
pointer, void, 74
POJ judge, xxix
pop in stack, 47
priority queue, 298
probability, 384
   multiplication rule, 384
   subtracting, 384
problem description
   components, xxx–xxxi
   input, xxx
   output, xxxi
   problem of, xxx
   subtask, 268
programming judge, xxviii
push in stack, 47

Q
qsort function for sorting, 10–11, 74
quadratic-time algorithm (O(n^2)), 10, 23, 410
querying the segment tree, 313–316
queue, 50
   first-in, first-out, 50
Quicksort, 398–400
   pivot, 398

R
rand function, 378–379
random hashing, 389
randomized algorithm, 389
   Las Vegas algorithm, 388–389
   Monte Carlo algorithm, 387–388
randomization, 377, 397–398
random numbers, 378
range maximum query, 307–308
range sum query, 262
   one dimension, 262–263
   two dimensions, 263–266
Rawlins, Gregory J.E., 402
reading integers
   from line, 106
   from string, 62–63
reading lines, 188–189
recursion, 52, 54, 65
   base case, 55
   recursive call, 54
   recursive case, 55
reflexive in union-find, 353
representative in union-find, 340–341
reversed in graph, 208
right child in binary tree, 40
   See also binary search
comparison function for
   sorting, 254
feasible solution, 246
greedy algorithm, 244, 247
invariant, 252
Roberts, Eric, 76
root in tree, 40
   See also graph
0-1 breadth-first search, 186
modeling as graph, 180–182
state in graph, 182
weighted in graph, 182
Roughgarden, Tim, 36, 195, 275
runtime
   of binary search, 241–242
   of breadth-first search, 172–173
   of Dijkstra’s algorithm, 210,
   420–421
   of heap, 290
   of segment tree, 315
   of union-find, 347, 349–350, 352
S
segment tree, 308, 311–312, 317
   height of, 309
   maximum index, 312–313
   maximum sum of two elements,
   319–323
   querying, 313–316
   runtime, 315
   updating, 324–325
shortest paths
   in graph, 172
   number in graph, 224
siblings in tree, 40
single-source shortest-paths
   counting edges, 172
   in general, 210
Social Network problem, 332–352.
   See also union-find
modeling, as graph, 333
solve function, xxxiii, 85, 101
sorting
   comparison function, 10–11, 74,
   254, 305
   heapsort, 299
   qsort function, 10–11, 74
   Quicksort, 398–400
South African Programming Olympiad
(SAPO), xxix
Spelling Check problem, 29–35
   incremental hash function, 31
   longest common prefix, 32
   of strings, 33
   longest common suffix, 32
   of strings, 33–34
SPOJ judge, xxix
srand function, 387
stack, 47
   implementation, 47–49
   last-in, first-out, 47
   pop, 47
   push, 47
   top, 47
standard input, xxxi
standard output, xxxi
state in graph, 182, 217
static keyword, xxvi–xxvii, 9
strncpy function, 75
subtask of problem description, 268
subtrees in tree, 40
Supermarket Promotion problem,
   277–298. See also heap
   complete
      in binary tree, 282
      in heap, 282
      extract-max, 280
      extract-min, 280
   max-heap, 282–283
      as array, 291
      extracting from, 287–289
      height of, 290
      inserting into, 283–287
      order, 283
   min-heap, 293
      extracting from, 294
      inserting into, 294
      order, 294
   runtime of heap, 290
symmetric in union-find, 353
T
Tardos, Éva, 123
ternary operator, 423
The Jumper problem, 125–137.
See also dynamic programming
solutions
backward, 134
forward, 134
subproblems
“exactly,” 128
order of in dynamic programming, 137
parameters, 128
Thinking Recursively with Java
(Roberts), 76
TLE (Time-Limit Exceeded) status,
xxx, 410
top-down solution, 97
top in stack, 47
transitive in union-find, 353
treap, 300–301
tree, 40
binary, 41
children, 40
descendant, 40, 66
edge, 40
height, 40
leaf, 40
node, 40
node struct, 68
parent, 40
root, 40
siblings, 40
subtrees, 40
vertex, 40
Two Sum problem, 318–329.
See also segment tree
maximum sum of two elements in segment tree, 319–323
updating the segment tree, 324–325
U
undirected in graph, 170
union-find, 341, 353
augmenting, 356
equivalence relation, 353
find operation, 341–342
inverse Ackermann function, 352
path compression, 351, 422
reflexive, 353
representative, 340–341
runtime, 347, 349–350, 352
symmetric, 353
transitive, 353
union by size, 348, 353, 367
union operation, 340, 343
Unique Snowflakes problem, 1–17.
See also hash table
collisions, 14
comparison function for sorting, 10–11
identical integers, checking for, 4
linked list, adding to, 15
node struct in linked list, 14
qsort function for sorting, 10–11
snowflake code, 12–13
updating the segment tree, 324–325
USA Computing Olympiad
(USACO), xxix
UVa judge, xxix
V
vertex in tree, 40
void pointer, 74
W
WA (wrong answer) status, xxx
Ways to Build problem, 137–149.
See also dynamic programming
memoization, 142
subproblems
adding, 145
“exactly,” 140
parameters, 140
weighted in graph, 182, 199
Y
Yōkan problem, 376–387. See also
randomized algorithm
binary search, 379–381
flavor array, 379
probability, 384–385
rand function, 378–379
randomization, 377
random numbers, 378
srand function, 387