

Index

SYMBOLS

- % (modulo operator), 86**
- & (AND) bitwise operator, 145**
- ^ (XOR) bitwise operator, 145**
- | (OR) bitwise operator, 145**
- << (left shift) operator, 145, 146**
- >> (right shift) operator, 146**
- >>> (logical shift right) operator, 146, 156**
- ~ (NOT) unary operator, 145**

A

- abstract class, 125**
- accomplishments, importance in developing marketable skills, 5**
- ACID compliance, 136**
- action verbs on a resume, 208**
- actions, 121. See also methods**
- acyclic list**
 - defined, 50
 - moving pointers at different speeds, 51–52
- add method, synchronizing with remove, 115–116**
- aggregates**
 - as a commonly used SQL feature, 135
 - returning the maximum value without, 139–140
 - using SUM, 138
- AJAX**
 - described, 186
 - rise of, 17

algorithm optimizations. See optimizations

algorithms

- classifying by relative efficiency, 20
- comparing the predicted relative performance of, 21
- focusing on for an interview problem, 18
- generating words from a telephone number, 101–103
- improving by focusing on a deficiency, 82

alphabetical order, listing permutations in, 95

already-encountered list, putting nodes into a separate, 50–51

ambiguity, eliminating, 129

ambitions, discussing, 194–195

ancestor

- finding the lowest common, 64–66
- of a node, 55

AND operation, solving Number of Ones, 157

API (application programming interface), 125

application developers, 1

arms-length recursion, 96, 99

array and string problems, 73–88

Array object in JavaScript, 70–71

arrays

- compared to hash tables, 74–75
- deleting elements from, 77
- described, 67–68
- processing long ASCII strings, 78
- providing constant-time lookup, 74
- rearranging data in, 77

assumptions

- identifying false, 161
- inherent in problem solving, 160
- looking for false, 167

asymptotic running time, 21
atomicity of database transactions, 136
attributes, 121
autobiography, resumes as, 206, 212
average case running time, 22

B

balanced tree, 57
bandwidth, 189
base cases
 invoking directly without a recursive call, 96
 in recursive algorithms, 89
 separating from recursive cases, 96
base class in inheritance, 122
base conversion problem, compared to Number of Ones, 155
best case running time, 22
BFS (breadth-first search), 46, 59
big companies, working for, 2
big-endian machine, 153
Big-endian or Little-endian problem, 153–155
big-O run-time analysis
 defined, 20
 described, 21–22
 example of, 20–21
 general procedure for, 22–23
 optimizations and, 23
binary operators, 145
binary representation of decimal digits, 85–86
binary search, 93
Binary Search problem, 92–95
binary search trees (BSTs)
 compared to hash tables, 190–191
 described, 56–58
 performing a preorder traversal of, 61–62
 performing lookups in, 57–58
 special properties of, 65
binary tree problems, 61–66
binary trees, 55–56
binary two's complement notation, 144–145
bit manipulation, as an interview topic, 144–146
bitwise operators, 145–146
Boat and Dock problem, 174–176
bookmarking sites, 3

bookstores, as a job market information source, 3
boundary conditions, checking code for, 19
boys, escaping a train, 183–184
brainteaser problems, 162–172
brainteasers
 described, 159
 drawing pictures to solve, 173
 graphical or involving spatial thinking, 173–184
 not being intimidated by, 161
 tackling, 159–162
breadth-first search (BFS), 46, 59
Bridge Crossing problem, 165–168
BSTs. See binary search trees
buffer, producers and consumers using simultaneously, 116–117
bug-finding problems, generic strategy for, 40
bugs, finding and fixing, 1
Bugs in removeHead problem, 40–41
bulleted lists on a resume, 208, 209
Burning Fuses problem, 180, 182
business cards, collecting from interviewers, 12
busy waiting, avoiding, 112–117
Busy Waiting problem, 112–114
buzzword compliance of resume information, 207
bytes, referring to individual characters, 71

C

C
 deleting an element in a linked list, 29
 endianness not specified, 154
 linked lists problems, 25
 next pointer bound with data, 26
 passing variables, 41
 pointer misuse in, 28
 removing all elements from a linked list, 30
 reversing the order of words in, 79
 shifted negative number as positive or negative, 146
 strings and character arrays as essentially identical, 67
 strings in, 71
 treatment of arrays, 68–69
C#
 classes for a tree of integers, 53–54
 coding a BST search in, 57
 creating generic linked lists, 26
 disallowing multiple inheritance of classes, 128–129

- function deleting characters from a string, 76–79
- garbage collection, 125, 188
- nested classes, 187
- shift operators, 145
- shifted number as negative, 146
- strings in, 73
- updating the reference to the head of the list, 27–28
- use of, 53
- C++**
 - arrays in, 70
 - bound with data, 26
 - code for stack implementation, 34–35
 - defining an interface, 125–126
 - deleting an element in a linked list, 29
 - differences from Java, 186–187
 - friend classes in, 187
 - linked list problems, 25
 - pointer misuse in, 28
 - removing all elements from a linked list, 30
 - shift operators, 145
 - shifted negative numbers, 146
 - strings and character arrays as essentially identical, 67
 - strings in, 72
 - treatment of arrays, 68–69
- C++ versus Java problem, 186–187**
- call stack, placing data on, 63**
- capabilities. See actions**
- career goals, discussing, 196**
- ceiling function, 172**
- certification, 4**
- char array in C, 71**
- char to Character mapping, 76**
- char type, holding 16-bit Unicode characters in Java, 72**
- character(s)**
 - converting into an integer, 74
 - deleting from a string, 76–79
 - described, 71
 - using as the index, 74
- character pointer, examining integer bytes, 154**
- character-to-numeric-value conversion, 84**
- child lists, 46–47**
- child node**
 - described, 55
 - separating each from the node before it, 48
- child pointers, 45**
- chronological ordering in a resume, 210**
- circle**
 - mathematical function producing, 147–148
 - radius and circumference in a brainteaser, 179–180, 181
- circular references in garbage collection, 189**
- circularly-linked lists, 27**
- classes**
 - defined, 121
 - implementing interfaces, 125, 126
 - passing via inheritance, 188
- classified ads as a job search method, 9**
- Codd, E.F., 131**
- code**
 - for all error and special cases, 19
 - interviewer questions about, 19
 - tracing through with an example, 19
 - written in the interview, 17
- coding questions**
 - as the meat of an interview, 15
 - problems in, 16
 - process followed in interviews, 15–17
 - scenario for answering, 15
 - solving, 17–19
- columns in tables, 131**
- combinations, recursive method for generating, 98**
- Combinations of a String problem, 97–100**
- combinatorial mathematics, 101**
- committing transactions, 136**
- common key, joining tables on, 134**
- companies**
 - contacting directly, 8–9
 - finding and contacting, 7–9
- Company and Employee Database problem, 137–138**
- company recruiters. See recruiters**
- CompareToAll**
 - big-O analysis of, 21, 22
 - implementation, 20–21
- CompareToMax**
 - big-O analysis of, 21, 22
 - implementation, 20
- compensation package, negotiating, 13**

- computer science students, languages and technologies required of, 3**
- computers, almost religious attachment to, 195**
- concurrency, 107**
- concurrency problems, 112–119**
- consistency of database transactions, 136**
- constant pointer, 68**
- constant running time, 22**
- constructing an object, 124**
- constructor method in a class, 124**
- Consumer thread writing with a Producer thread, 114–117**
- content organizing on a resume, 208–209**
- Count Open Lockers problem, 163–164**
- count values for each character in a string, 74**
- Counting Cubes problem, 176–179**
- counting semapores, 108**
- createStack function, 33**
- createStack routine, 33**
- credentials, upgrading, 4**
- cryptography algorithm, discovery of a new, 190**
- Cryptography problem, 189–190**
- cubes**
 - counting in layers, 177
 - counting those not on the surface, 177
 - defined, 178
 - faces of, 176
- cubic array**
 - counting cubes on the surface, 177
 - cubes on the surface of, 176–179
- current position pointer, 42**
- curriculum vitae, compared to a resume, 207**
- cycle avoidance, in a circularly-linked lists, 27**
- cyclic list**
 - defined, 50
 - moving pointers at different speeds, 51–52

D

- data elements. See nodes**
- data structure**
 - restoring to its original condition, 48–49
 - trying a different, 19
- database problems, 136–141**
- database transactions**
 - described, 135–136
 - properties of, 136

- databases, fundamentals, 131–136**
- deadlocks**
 - described, 109
 - determining occurrences of, 117–119
 - solution to breaking, 119
- debuggers, 1**
- default implementation, of an interface, 126**
- deleteStack function, 33**
- deleteStack operation, 34**
- deletions**
 - in BSTs, 58
 - from a linked list, 29–30
 - in a linked list, 36
- DeMorgan's Law, 152**
- denominator, 176**
- depth-first search (DFS), 59**
- derivative, ratio of rates of change between two variables, 174, 175**
- descendants of a node, 55**
- design-oriented jobs, less likely to be outsourced, 4**
- destination position, tracking for the write position, 77**
- destructors in C++, 124–125**
- developer job market. See job market**
- DFS (depth-first search), 59**
- diagrams, importance of drawing to solve puzzles, 173–174**
- Dice job site, 3, 8, 9**
- digit characters, value of, 84**
- The Dining Philosophers problem, 117–119**
- directed graph, 60**
- DISTINCT keyword with SELECT, 140**
- “Do You Have Any Questions for Me?”, 199**
- doubly-linked lists**
 - described, 27
 - flattening, 44–47
- dress for interviews, 10–11**
- duck, maximizing the head start of, 180**
- durability of database transactions, 136**
- dynamic arrays**
 - defined, 31
 - implementing a stack, 31
 - in JavaScript, 70–71
 - as a language feature, 68
- dynamic data structures, compared to arrays, 68**

E

edges in graphs, 60
Eighth of a Circle problem, 146–148
 electronic devices, turning off before
 interviews, 11
 element list, as a special case, 36, 38
 elements
 finding based on position, 42
 inserting and deleting in a singly-linked list, 29–30
 Ellipse subclass in Shape, 123–124
 empty list, as a special case, 36, 38
 empty or null link in a linked list, 26
 encapsulation, 122
 encapsulators, interfaces as, 125
 end node, special properties in a cyclic list, 50
 endian, familiarity with the term, 153
 endianness, 153, 154
 equality (=) operator, testing for a NULL column
 value, 141
 error code arguments, 32–33
 error codes, returned by `createStack` and
 `deleteStack`, 33
 error conditions
 checking, 41
 checking common, 40
 watching for, 42
 error correction in graphics algorithms, 143
 error handling for `push` and `pop` operations, 32
 Escaping the Train problem, 183–184
 estimation problems, 162
 event semaphores, 108
 event thread, 108
 blocking, 114
 examining input items in big-O analysis, 21
 examples
 going back to, 19
 starting an interview problem with, 18
 working through, 161
 exception throwing, pros and cons of, 94
 experience
 answering questions about, 193–194
 discussing, 195
 exploding signing bonus, 12
 exponents, expressing repeated multiplication, 171
 extends. See inheritance

F

faces of a cube, 176
 factorial, recursive implementation of, 89–91
 factorial operator ($n!$), 89
 factors, using to solve Count Open Lockers,
 163–164
 filmstrip, visualizing time as, 178
 finalizer method, 125
 Find the First Nonrepeated Character problem,
 73–76
 first element case, writing for `remove`, 36
 fit, as a key theme of nontechnical questions, 194
 flattened list, creating, 48
 floor operation of rounding, 164
 food chain. See programmer food chain
 foreign keys, 132, 133
 formatting resumes, 208
 4D (four-dimensional) hypercubes, 178
 The Fox and the Duck problem, 179–180, 181
 fractional weighing, counting as full
 weighing, 172
 friend classes, in C++, 187
 Friend Classes problem, 187
 function
 checking data coming into, 40
 checking each line of, 40–41
 returning the maximum value stored in an array, 20–21
 returning values properly, 40, 41
 fuses, lighting both ends of, 182

G

garbage collection
 described, 188–189
 linked lists and, 26
 Garbage Collection problem, 188–189
 garbage collector, finding and destroying unused
 objects, 125
 general middle case, writing for `remove`, 36–37
 general solutions, interview problems with, 82
 generic tree, compared to a binary tree, 56
 geometric drawing, converting to a pixel-based
 raster image, 147
 geometric equation, translating to a pixel
 display, 143

goals, discussing, 194–195
grades in school, 5
graduate degrees, upgrading credentials with, 4
graphical and spatial problems, 174–184
graphical and spatial puzzles, 173–184
graphics and bit operations, problems on, 146–157
graphics, as an interview topic, 143–144
graphs, 60–61
GROUP BY clause with SELECT, 135
GROUP BY feature, using, 138

H

hash tables
benefit of, 188
versus binary search trees, 190–191
compared to arrays, 74–75
implementing for Unicode strings, 75–76
providing constant-time lookup, 74
Hash Tables versus Binary Search Trees problem, 190–191
head element
in a doubly-linked list, 27
removing from a singly-linked list, 40–41
of a singly-linked list, 26
tracking in a singly-linked list, 27–28
head pointer, updating, 36
headhunters, working with, 8
heaps, 58–59
Heavy Marble problem, 169–172
highest-order term used in big-O analysis, 21
hiring managers, more flexible than recruiters, 11
Horner's Rule, 84
HotJobs, 3
“How Much Money Do You Want to Make?”, 196–198
human resources representative. See recruiters
hypercubic arrays of hypercubes, 178–179

I

if statements, removing by loop partitioning, 100
immutable strings
in C#, 73
in Java, 72

implicit typing in JavaScript, 73
infinite recursion, 90
information sources on the job market, 3
inheritance
classic example of, 123–124
defined, 122
Inheritance problem, 187–188
inorder traversal of a node, 60
INSERT SQL statement, 133, 137
insertAfter function, writing, 39–40
insertion
in BSTs, 58
in a linked list, 36
integers
accessing bytes of, 154
binary search on a sorted array of, 92–95
comparing without using comparative operators, 16
converting a signal back into, 85–88
Integer/String Conversion problem, 83–88
integer-to-string routine, 85–88
integrity of data stored in a database, 135
interactivity in the interview, 17
interfaces
default implementation of, 126
defined, 125
as encapsulators, 125
mitigating the restriction of single inheritance, 129
Interfaces and Abstract Classes problem, 125–126
internships, value of, 5
interview database problems, 132–133
interview problems
with less-obvious special-case solutions, 82
steps in solving, 18–19
interview process, stages of, 9–11
interviewers, collecting business cards from, 12
irrelevant items, on a resume, 209
IS NOT NULL syntax with SELECT, 141
IS NULL syntax with SELECT, 141
isolation of database transactions, 136
iterative algorithms
as often easy to write, 91
replacing recursions, 63
iterative alternative to a recursive algorithm, 103–105

iterative analog of the recursive binary search, 94–95
iterative implementation of the lowest common ancestor traversal, 66
iterative routines, using looping constructs, 91
iterative solutions, as usually more efficient than recursive solutions, 91

J

jagged array in C#, 70

Java

classes for a tree of integers, 54–55
 coding a BST search in, 57
 creating generic linked lists, 26
 defining an interface, 126
 differences from C++, 186–187
 disallowing multiple inheritance of classes, 128–129
 garbage collection, 125, 188
 nested classes, 187
 nonstatic methods as always virtual, 127
 performing sign extension when shifting right, 146
 shift operators, 145
 strings in, 72–73
 updating the reference to the head of the list, 27–28
 use of, 53
 using a fixed endianness, 154

JavaScript

arrays in, 70–71
 changed industry view of, 17
 performing sign extension when shifting right, 146
 shift operators, 145
 strings in, 73

job application process, 7–14

job descriptions on a resume, 208–209

job fairs as a job search method, 9

job interviews, preparation for, 201

job market, understanding, 3

job openings, targeting specific, 9

job sites

described, 9
 online, 3

jobs, reasons for changing, 196

joining tables, 134, 138

K

keeping learning, 5

kernel-level threads, 108

keys in tables, 131–132

knowledge-based questions during interviews, 185–191

L

languages

almost religious attachment to, 195
 knowledge of mainstream, 16
 less-commonly used or more-advanced aspects of, 19
 less-mainstream, 16
 used in coding questions, 16–17

last element case, writing for remove, 37

last-in-first out (LIFO) data structure, a stack as, 30

latency, 189

learning, continuing, 5

least-significant byte (LSB) to most-significant byte (MSB), 153

leaves, 55

left child in a binary tree, 55

length method in Java, 72

length property of a JavaScript array, 71

letters

corresponding to numbers on a telephone keypad, 100
 marking as used or unused, 96

library routines for string/integer conversions, 83

line segment, drawing a, 143–144

linear running time, algorithms with, 78

linear singly-linked list, 26

linear time in big-O analysis, 21

line-by-line analysis of a function, 40–41

linked list(s)

allocating memory dynamically for each element, 31
 compared to arrays, 67
 compared to dynamic arrays, 31
 kinds of, 25
 length of, 36
 less complicated to implement than dynamic arrays, 31
 making two complete traversals of, 42
 operations, 27–30
 problems, 25
 removing all elements from, 30
 routines, 26
 as shorthand for the first element of a linked list, 26

linked list typical problems, 30–52

LinkedIn, 9

List Flattening problem, 44–47

list length circumstances, potentially problematic, 38

List Unflattening problem, 48–49

little-endian machine, 153

livelock, 119

local copy, updating instead of the head pointer, 28

location of arrays, tracked in C and C++, 69

locomotives, smashing a bird, 176

logical && and || operators, 145

logical operations, three possible values of in SQL, 141

long-term projects, compared to short-term, 2

lookup operations in a BST, 57–58

loop index dependent conditionals, 100

loop partitioning, 100

Lowest Common Ancestor problem, 64–66

LSB. See least-significant byte

M

m element data structure, implicitly advancing, 42

Maintain Linked List Tail Pointer problem, 35–40

management, as a goal, 2

management job, taking, 4

managers, resumes for, 212–218

mark and sweep garbage collection, 189

market. See job market

marketable skills, developing, 4–5

marketing tool, resume as, 4, 206

mask algorithms, comparing, 156

masks, getting the value of any bits, 155–156

Max, No Aggregates problem, 139–140

max value, extracting with a heap, 58–59

maximum value

function returning, 20–21

returning using SQL without an aggregate, 139–140

m-behind pointer

with a current position pointer, 42

explicitly advancing, 42

memory deallocation by garbage collection, 188

memory requirements of arrays versus hash tables, 74–75

methods. See also actions

add method, 115–116

constructor method in a class, 124

finalizer method, 125

length method in Java, 72

nonvirtual in C# and C++, 127

remove method, 115–116

modulo of a negative number, 86

modulo operator (%), 86

monitors

around code using or altering the value of
userBalance, 111

described, 108–109

Monster, 3, 9

most-significant byte (MSB), 153

MSB. See most-significant byte (MSB)

Mth-to-Last Element of a Linked List problem, 41–44

multibyte data type, determining LSB and MSB, 154

multibyte encodings, 71

multidimensional arrays

implementing as linear, 67

in Java, 70

Multiple Inheritance problem, 128–129

multiple inheritance, simulating in Java using interfaces, 187

multithreaded applications, classical problems involving, 112–114

multithreaded programming, 107

mutable strings

in C++, 72

creating in Java, 72

creating with `StringBuilder` class in C#, 73

mutex semaphore, 108

N

native threads, 108

negative numbers

in binary two's complement notation, 144

character-to-numeric conversion of, 84

modulo of, 86

negotiating salary, 12–13

nested classes in Java and C#, 187

Network Performance problem, 189

networking performance, major issues in, 189

networking through a contact with a company, 7–8

New Cryptography Algorithms problem, 190

next pointer

- bound with data, 26
- in a linked list, 26

next reference in a linked list, 26

nodes, trees made up of, 53

nonbinary trees, traversals happening with, 60

non-negotiable factors in a negotiation, 198

non-negotiable offer, as a hardball negotiation tactic, 13

non-null return value, checking for in a linked list, 29

nonrecursive solution to a problem, 103–105

nonrepeated character, finding the first, 73–74

nontechnical questions

- in interviews, 193–199
- reasons for, 193–198

nonvirtual methods in C# and C++, 127

nonword characters, 79

notification mechanism in busy waiting, 113–114

NULL, compared to NULLCHAR in C, 71

null bytes, storing in C++ strings, 72

null character, marking the end of string in C, 71

NULL column value, testing for, 141

Null or Cycle problem, 49–52

NULL pointer arguments

- checking behavior for, 38
- as a problem-prone circumstance, 36, 38

NULL pointer in the last element case, 37–38

null pointers, storing, 32

NULL-terminated code, 50

Number of Ones problem, 155–157

numbers

- letters corresponding to on a telephone keypad, 100
- with unpaired factors, 164

numerator, 176

O

object-oriented (OO) languages, 34, 121

object-oriented programming. See OO programming

objects

- defined, 121
- as instances of classes, 124
- number of possible arrangements of, 96

obvious answer, almost never correct for a brain-teaser, 160

off-by-one errors, introducing, 43–44

offers

- accepting and rejecting, 13–14
- deciding on, 12
- increasing when less than expected, 198

offshoring, 3

one-element list

- checking, 41
- checking behavior for, 38

one-way edges, graphs with, 60

online job listings, 3

online profile, sanitizing, 5–6

on-site interviews, 10

OO (object-oriented) programming, 121–129

open-source development project, starting or joining, 4

open-source projects, 2

operating systems, almost religious attachment to, 195

optimizations, big-O analysis and, 23

optimizing, moving a group of items a few at a time, 168

order of magnitude calculation, doing a rough, 162

out-of-bounds array accesses, identifying in C/C++ programs, 69

output position, tracking, 99

outsourcer, working for, 4

outsourcing

- avoiding jobs headed for, 3–4
- rise of, 3–4

overlap

- testing for, 151–152
- ways rectangles can, 150–151

P

parent class, 122

parent node, 55

parents of nodes in a graph, 60

pattern-based approach to multiple dimensions, 179

patterns, examining a list of permutations for, 96

PDA (personal digital assistant), address book structure for, 190–191

PDF file, resume in, 9

perfect squares, counting, 164

permutation process, defining, 96

permutations, going through in a systematic order, 95

Permutations of a String problem, 95–97

persistence in finding a correct solution, 19

personal digital assistant (PDA), address book structure for, 190–191

pickup order, one change breaking the deadlock, 119

pictures, drawing to solve brainteasers, 173

pipe, as an analog for a network, 189

pixel density, in a line-drawing algorithm, 144

pixel-based raster image, converting a geometric drawing to, 147

pixels, algorithms changing the colors of, 143

place value, determining for each character digit, 84

Point class definition, 121–122

pointer(s)

advancing at different speeds, 51–52

passing to a variable, 32, 41

required for deletions from a linked list, 30

to a stack, 31

understanding, 25

pointer constant, 68

pointer misuse, C/C++, 28

points, falling inside a rectangle, 150

polymorphism

classic example of, 123–124

defined, 122

virtual methods used for, 127–128

pop operation

coding for, 34

possibilities for the interface to, 32

returning an error code, 33

on a stack, 30, 31

positive integers, as square roots of perfect squares, 164

postorder traversal of a node, 60

preemptive threading, 108

preorder traversal

of a binary search tree without using recursion, 62–64

coding using recursion, 62

of a node, 59

Preorder Traversal, No Recursion problem, 62–64

Preorder Traversal problem, 61–62

preparation

for job interviews, 201

for knowledge-based questions, 185

primary key, 131

adding, 133

primes, unique factor properties of, 163

printing of resumes, 210

problems

beware of simple, 161–162

Big-endian or Little-endian, 153–155

Binary Search, 92–95

Boat and Dock, 174–176

breaking into parts, 161

Bridge Crossing, 165–168

Bugs in removeHead, 40–41

Burning Fuses, 180, 182

Busy Waiting, 112–114

C linked list, 25

C++ linked list, 25

C++ versus Java, 186–187

Combinations of a String, 97–100

Company and Employee Database, 137–138

Count Open Lockers, 163–164

Counting Cubes, 176–179

Cryptography, 189–190

The Dining Philosophers, 117–119

Eighth of a Circle, 146–148

Escaping the Train, 183–184

Find the First Nonrepeated Character, 73–76

The Fox and the Duck, 179–180, 181

Friend Classes, 187

Garbage Collection, 188–189

getting stuck on, 19

Hash Tables versus Binary Search Trees, 190–191

Heavy Marble, 169–172

Inheritance, 187–188

Integer/String Conversion, 83–88

Interfaces and Abstract Classes, 125–126

List Flattening, 44–47

List Unflattening, 48–49

Lowest Common Ancestor, 64–66
 Maintain Linked List Tail Pointer, 35–40
 Max, No Aggregates, 139–140
 Mth-to-Last Element of a Linked List, 41–44
 Multiple Inheritance, 128–129
 Network Performance, 189
 New Cryptography Algorithms, 190
 Null or Cycle, 49–52
 Number of Ones, 155–157
 Permutations of a String, 95–97
 Preorder Traversal, 61–62
 Preorder Traversal, No Recursion, 62–64
 Rectangle Overlap, 149–153
 Remove Specified Characters, 76–79
 Reverse Words, 79–83
 Simple SQL, 136–137
 Stack Implementation, 30–35
 Telephone Words, 100–105
 Three Switches, 164–165
 Three-Valued Logic, 140–141
 Virtual Methods, 127–128
problem-solving ability, using brainteasers to assess, 159
problem-solving process in interviews, 17
Producer thread, writing with a Consumer thread, 114–117
Producer/Consumer problem, 114–117
professional development courses, 3
programmer, determining what kind you are, 1–2
programmer food chain, working up, 4
programming, reading about, 201
Programming Interviews Exposed mailing list, joining, 201
programming job, finding the kind you enjoy, 1–2
programming problems, approaches to, 15–23
projects, types of, 2
proofreading resumes, 210
properties. See attributes
proprietary, closed-source projects, 2
pseudo-events, queueing for processing by the event thread, 114–117
public key cryptography
 compared to symmetric key, 189–190
 described, 189, 190
public keys over insecure channels, 190

public profile. See online profile
public static function in Java or C#, 75–76
push operation
 coding for, 33
 on a stack, 30
 taking a data argument and returning an error code, 32
push routine for a stack, 31
puzzles, training your mind, 201
Pythagorean theorem, 175

Q

quality assurance (QA), 2
query, fetching data, 131
questions. See also coding questions
 asking factual, 18
 for the interviewer, 199

R

raster pixel display, 143
real-world problems, modeling with graphs, 61
recruiters
 drawing attention away from negative aspects, 11
 high-pressure tactics, 12
 role of, 11–12
 signing you at lowest possible salary, 12
 territory of some, 12
Rectangle Overlap problem, 149–153
Rectangle subclass in Shape, 123–124
rectangles, conditions for not overlapping, 152–153
recursion
 implementing a traversal, 60
 subtree property as conducive to, 58
 understanding, 89–92
 using, 48
 using stack data structure, 63
recursion problems, 92–105
recursive algorithms
 implementing without using recursive calls, 92
 replacing with iterative algorithms, 63
recursive calls, eliminating the need for, 92
recursive cases, 89
recursive definition of a preorder traversal, 62
recursive implementation of a binary search, 93

recursive preorder traversal, emulating iteratively, 64

recursive routines, 89

recursive solution, code for, 49

red-black tree, 58

reference counting in garbage collection, 188–189

references

- explicitly disambiguating, 129
- for headhunters, 8
- not mentioning on a resume, 209
- storing in Java or C#, 26

referential integrity, 132

relational databases, 131–132

relevant information, including on a resume, 209

remove function, writing, 36–39

remove method, synchronizing with add, 115–116

Remove Specified Characters problem, 76–79

resumes

- checking for viruses, 9
- described, 203
- examples of good, 218–221
- examples of technical, 203–218
- formatting, 208
- keeping short, 206–207
- reviewing prior to interviews, 185
- sending as plain text in the body of the e-mail, 9
- submitting through the Internet, 8–9

reverse string function, designing, 82

Reverse Words problem, 79–83

right child in a binary tree, 55

rolling back transactions, 136

root node

- as an ancestor to all nodes in a BST, 65
- of a heap, 58

rounding

- in graphics algorithms, 143
- in a line-drawing algorithm, 144

routines

- indicating success or failure of, 32
- recursive, 89

rows in tables, 131

Rubik's Cube, 176

rules of coding questions, 16

run-time analysis

- of algorithms, 20
- fastest-possible running time for any, 22

S

salary experience, discussing, 196–198

salary history, discussing, 199

salary, negotiating, 12–13

salary range, obtaining from the interviewer, 197

salary review, having in six months, 198

scan conversion, 147

scanner. See token scanner

schema, 131, 132–133

screeners, examining prospective job applicants, 5

screening interviews, 9–10

search algorithms, searching for a particular node, 59

searches, common invoking trees, 59

security by obscurity, 190

SELECT SQL statement, 133–135

selling yourself in a resume, 206

semaphores

- avoiding busy waiting, 113
- described, 108–109
- kinds of, 108

senior developers, resumes for, 212–218

senior position, stressing management skills and experience, 212

Shape class, 123

shapes library for a vector-based drawing application, 123

shared key cryptography. See symmetric key cryptography

shared resources, concurrency issues involving multiple, 117

shift operators, 145, 146

short-term projects, compared to long-term, 2

side project, working on, 4

sign bit in binary two's complement notation, 145

- sign extension, when shifting right, 146**
- signing bonus, exploding, 12**
- Simple SQL problem, 136–137**
- single inheritance**
 - limiting classes to, 128
 - as restrictive, 129
- single mask algorithm, compared to multiple, 156**
- single-threaded coding, 107**
- singly-linked lists, 25–26**
- small company, working for, 2**
- social networking sites, 3, 7**
- software architect, 2**
- software development firms, working for, 4**
- software development, offshoring of, 3**
- solutions**
 - analyzing, 20–23
 - explaining to the interviewer, 18
 - obvious and less-obvious to interview problems, 82
 - wrong occurring first to most people, 172
- source position, tracking for the read location, 77**
- spatial problems, 174–184**
- spatial visualization problems, 176**
- special cases**
 - exercises on, 39–40
 - identifying, 36
- special-case solutions, interview problems with less-obvious, 82**
- specific answers, preferred to knowledge questions, 186**
- SQL (structured query language)**
 - highlights of, 132–135
 - use of ternary logic, 141
- sqrt function, using, 148**
- Stack Implementation problem, 30–35**
- stack-based interactive routine, implementing, 92**
- stacks**
 - described, 30
 - iterative implementation for storing data on, 63–64
 - as a last-in-first-out data structure, 64
- states. See attributes**
- static arrays**
 - Java arrays as, 69
 - in most dynamic array implementations, 68
- strcpy function in C, 71**
- string class, immutable in C#, 77**
- String class in C#, 73**
- string class in C++, 72**
- String class in Java, 72**
- string function, designing an in-place reverse, 82**
- string problems, 73–88**
- StringBuffer class in Java, 72**
- StringBuilder class in Java, 72**
- strings**
 - as closely related to arrays, 67
 - converting signed integers back into, 85–88
 - converting to signed integers, 83–85
 - described, 71–73
 - printing all possible combinations of characters in, 97–100
 - printing all possible orderings of the characters in, 95–97
 - reversing in place by exchanging characters, 82
 - reversing the order of words in, 79–83
 - storing internally as arrays, 71
 - truncating in C, 71
- string-to-integer routine, converting a string to a signed, 83–85**
- strlen function in C, 71**
- struct, declaring for a linked list element, 31**
- structured query language. See SQL**
- subclass in inheritance, 122**
- subquery, using, 138**
- subtasks, tasks defined in terms of similar, 89**
- subtree property, as conducive to recursion, 58**
- subtrees, thinking in terms of, 61–62**
- suit, as overkill for a technical job interview, 11**
- SUM aggregate, using, 138**
- symmetric key cryptography, compared to public key cryptography, 189–190**
- synchronized keyword, creating a monitor in Java, 110–111**
- system programmers, 1**
- system threads**
 - described, 108
 - versus user threads, 108–109
- System.array abstract base class in C#, 70**

T

tables

- adding values to, 133
- joining, 134, 138
- in a relational database, 131
- retrieving data from, 133–135
- retrieving data from two, 138

tail element

- in a doubly-linked list, 27
- of a singly-linked list, 26

tail pointer, maintaining, 36

tail recursion, 90

talking, while solving coding questions, 17

team

- fitting in with existing, 194
- wanting to work with a great, 195

technical career paths, companies with, 2

technical interview sites, visiting, 201

technical interviews, nontechnical questions in, 193–199

technical questions, examples of, 194–199

technical resumes, examples of, 203–218

technical skills, categorizing by type, 207, 208

Telephone Words problem, 100–105

“Tell Me About Your Experience”, 195

temporary buffer, eliminating the need for, 82

temporary storage data structure, using to traverse a list, 42

temporary string buffer, allocating for a modified string, 77

ternary logic, 141

testing, 2

themes, shared by brainteasers, 159

thinking outside the box, questions identifying, 165

thought processes, working through a programming problem, 17

thread synchronization, constructs of, 108

threading, example of, 109–112

threads

- described, 107–108
- implementing, 107–108
- incorrect use of, 107

Three Switches problem, 164–165

three-dimensional problems, solving, 173–174

three-valued logic. *See* ternary logic

Three-Valued Logic problem, 140–141

time, representing a fourth dimension, 178

timeout, adding for deadlocks, 119

tooggling a locker, 163

token scanner, 79

transactions. *See* database transactions

transfer, as two operations, 136

travelers, crossing a bridge, 165–168

traversals, 59–60

traversing elements of a linked list, 28–29

tree-related vocabulary, 55

trees, 53–55

two-element list, as a special case, 36, 38

two-element lists, checking behavior for, 38–39

two-way pointers, graphs with, 60

U

undirected graph, 60

- modeling real-world problems, 61

Unicode characters, possible values of, 74

Unicode character type in Java and C#, 71

Unicode strings, code needed to process, 75–76

union types, 155

UNKNOWN value, 141

unpaired factors, numbers with, 164

unsigned integer, reading a value as, 156

unsorted lists, detecting, 93

user interfaces, coding, 1

user threads

- described, 108
- versus system threads, 108–109

V

version numbers, including in a resume, 208

vertical lines, accounting for, 143

virtual base classes, declaring, 129

virtual keyword in C# and C++, 127

virtual methods, advantages and disadvantages of, 128

Virtual Methods problem, 127–128

Visual Basic, avoiding in interviews, 17

visualization

appropriate for three-dimensional problems, 174

different techniques useful in, 176

extending into many dimensions, 178–179

of time, 178

void pointer storage, yielding a struct, 31

W

wchar_t (wide character) type in C++, 72

weighing items with a two-pan balance, 172

weighings, minimum number, 169–172

“What Are Your Career Goals?”, 196

“What Do You Want to Do?”, 194–195

“What Is Your Favorite Programming Language?”, 195

“What Is Your Salary History?”, 199

“What Is Your Work Style?”, 195

while loop, changing to a do...while loop, 87

“Why Are You Looking to Change Jobs?”, 196

“Why Should We Hire You?”, 199

word characters, 79

words

corresponding to a seven-digit number, 101

recognizing start and end, 79

reversing the order of in a string, 79–83

writing into a temporary buffer, 79–80

work style, discussing, 195

working solution of interview problems, 92

worst case running time, 22

wrapper routine, hiding array allocation and recursion level tracking, 91

Z

zero-element lists

checking, 41

checking behavior for, 38

