Index

A

adjacency 282

adjacency lists 287, 288

adjacency matrix 288-290

algorithm design techniques 57, 58
- divide-and-conquer 60, 61
- dynamic programming 68, 69
- greedy algorithms 74-76
- recursion 59, 60

algorithms 1, 35
- benefits 36
- criteria 36, 37
- example 37
- performance analysis 37, 38
- running time complexity, computing 52-54

amortized analysis 49
- accounting method 50
- aggregate analysis 50
- potential method 50

Anaconda distribution
- download link 5

AND operator 17

arithmetic expression 194
- infix notation 194
- postfix notation 194, 196
- prefix notation 194, 195

arrays 94, 248
- used, for implementing stacks 145-147

asymptotic notation 41
- Big O notation 44-47
- omega notation 47-49
- theta notation 42-44

B

balanced binary tree 184

base address 94

base cases 59

basic data types 7
- Boolean 8
- numeric 7, 8
- sequences 9
- tuples 18

Big O notation 44-47

binary heap 222
- implementing 223

binary search 61, 62

binary search algorithm 325-331

binary search tree (BST) 201, 273
- benefits 216-219
- example 201, 202
- maximum node 215, 216
- minimum node 215, 216
- nodes, deleting 209-214
- nodes, inserting 203-207
- operations 202
- tree, searching 208, 209

binary tree 181
- applications 194
- balanced binary tree 184
- complete binary tree 183

example 182
expression trees 194
full binary tree 182
nodes, implementing 184-186
perfect binary tree 183
regular binary tree 182
simple binary tree 181
unbalanced binary tree 184

bipartite graph 285

Boolean 8

Boyer-Moore algorithm 415
bad character heuristic 417-420
good character heuristic 420-424
implementing 424-426
working 416, 417

breadth-first search (BFS) 291-298

brute force algorithm 397-400

brute-force approach 58

bubble sort algorithms 346-352

bucket 252

C

ChainMap object 30

child node 181

circular linked lists 129, 130
creating 131
element, deleting 134-138
items, appending 131-133
querying 134
traversing 131

collections module 27
data types 27
operations 27

collisions
resolving 252, 253

command line
Python development environment, setting up via 3, 4

complete binary tree 183, 222

complex data types 19
dictionary 19, 20
set 23, 24

complexity classes
composing 50-52

complex number 8

container datatypes, collections module
ChainMap object 30
counter objects 31
default dictionary 29, 30
deque 28, 29
named tuples 27, 28
ordered dictionary 29
UserDict 32
UserList 32
UserString 33

counter objects 31

D

data structures 1

data types 6

default dictionary 29, 30

degree
of vertex/node 282

delete operation
implementing, in heap 229-233

depth-first search (DFS) 299-305

deque 28
functions 29

deterministic selection 383
implementation 386-393
working 384, 385

dictionary 19, 20, 248
- characteristics 21
- hash table, implementing as 263, 264
- methods 22, 23

directed acyclic graph (DAG) 284, 285

directed graph 283
- indegree 284
- isolated vertex 284
- outdegree 284
- sink vertex 284
- source vertex 284

divide-and-conquer design technique 60, 61
- binary search 61, 62
- merge sort 63-68

double hashing technique 267-271

doubly linked lists 114
- creating 115
- items, appending 116
- items, deleting 124-129
- node, inserting at beginning 116, 117
- node, inserting at end 119, 120
- node, inserting at intermediate position 121-123
- querying 123, 124
- traversing 115

dynamic programming 68
- bottom-up approach 70
- top-down with memoization 69

dynamic programming problems
- characteristics 69

E

edge 181, 282

elements
- retrieving, from hash table 260-262
- storing, in hash tables 257, 258

empty tree 181

exponential search algorithm 337-341

expression trees 194
- reverse Polish expression, parsing 196-200

F

factorial 59

Fibonacci series
- calculating 70-74

first in first out (FIFO) 157, 237

float type 8

frozenset 26

full binary tree 182

G

generator 99

graph methods 305
- Kruskal's Minimum Spanning Tree 306-309
- Minimum Spanning Tree (MST) 305, 306
- Prim's Minimum Spanning Tree 309-311

graph representations 286
- adjacency lists 287, 288
- adjacency matrix 288-290

graphs 281
- adjacency 282
- bipartite graphs 285
- degree of vertex/node 282
- directed acyclic graph (DAG) 284, 285
- directed graphs 283
- edge 282
- example 282
- leaf vertex 282
- loop 282
- node 282
- path 282

undirected graphs 283
vertex 282
weighted graphs 285

graph traversals 291
breadth-first search (BFS) 291-298
depth-first search (DFS) 299-305

greedy algorithms 74
examples 75, 76
shortest path problem 76-89

H

hashing functions 249, 250
perfect hashing functions 251, 252

hash tables 247, 248
elements, retrieving from 260-262
elements, storing 257, 258
example 248
growing 258-260
implementing 256, 257
implementing, as dictionary 263, 264
testing 262, 263

heap data structure 221
binary heap 222
binary heap example 223
delete operation 229-233
element, deleting at specific location 234, 235
heap sort 236, 237
insert operation 224-228
max heap 221
max heap example 222
min heap 222
min heap example 222

I

identity operators 16, 17

immutable sets 26

indegree 284

infix notation 194

in operator 15

in-order tree traversal 186-188

insertion sort algorithm 352-354

insert operation
implementing, in heap 224-228

integer data type 7

interpolation search algorithm 331-337

is not operator 17

isolated vertex 284

is operator 16

J

jump search algorithm 320-325

Jupyter Notebook
Python development environment, setting up via 4, 5

K

Knuth-Morris-Pratt (KMP) algorithm 406, 407
implementing 413-415
prefix function 408-410
working 410-413

Kruskal's Minimum Spanning Tree 306-309

L

last in first out (LIFO) 142, 145

last in last out (LILO) 142

leaf node 180

leaf vertex 282

level-order tree traversal 191-193

linear probing 254, 255

linear search 314, 315
ordered linear search 317-320
unordered linear search 315-317
linked list 287
linked list-based queues 163
dequeue operation 165, 166
enqueue operation 163-165
linked lists 95
circular linked lists 129, 130
doubly linked lists 114
nodes 95-98
pointers 95-98
practical applications 138, 139
properties 95
singly linked lists 98
used, for implementing stacks 148, 149
Linux-based operating system
Python, installing for 3
list-based queues 159
dequeue operation 161, 162
enqueue operation 159-161
lists 11, 12, 248
properties 12-14
logical operators 17, 18
loop 282

M

Mac operating system
Python, installing for 3
matrix 288
max heap 221, 222
membership operators 15
merge sort 63-68
min heap 222
example 222
implementing 224
Minimum Spanning Tree (MST) 305, 306
module 27

N

named tuples 27
negative indexing 19
nodes 95-98, 180, 282
not in operator 15
NOT operator 18
numeric types 7
complex 8
float 8
integer 7

O

objects 7
offset address 94
omega notation 47-49
open addressing 254
ordered dictionary 29
ordered linear search 317, 318
implementation 319, 320
OR operator 17
outdegree 284

P

parent-child relationship 179
parent node 181
path 282
pattern matching algorithms 397
Boyer-Moore algorithm 415
brute force algorithm 397-400
Knuth-Morris-Pratt (KMP) algorithm 406, 407
Rabin-Karp algorithm 401, 402

peek operation 154

pendant vertex 282

perfect binary tree 183

perfect hashing functions 251, 252

performance analysis, algorithm
space complexity 40, 41
time complexity 38, 39

pointers 95-98

pop operation 151
implementing, on stack 151-153

postfix notation 194, 196

post-order tree traversal 190, 191

prefix notation 194, 195

pre-order tree traversal 188-190

Prim's Minimum Spanning Tree 309-311

priority queue 221, 237
delete operation, implementing 241
demonstration 238
implementation 242-244
implementation, in Python 239
insertion operation, implementing 240
usage example 241

Priority Queue (PQ) 194

push operation 149-151

Python 1, 2
installing 2
installing, for Linux-based operating system 3
installing, for Mac operating system 3
installing, for Windows operating system 2, 3
references 1

Python 3.10 2

Python development environment
setting up 3
setting up, via command line 3, 4
setting up, via Jupyter Notebook 4, 5

Q

quadratic probing technique
for collision resolution 264-266

queues 157
applications 173-176
linked list-based queues 163
operations 158, 159
list-based queues 159
stack-based queues 166

quickselect algorithm 379
working 379-382

quicksort algorithm 359-369
working 378

R

Rabin-Karp algorithm 401
implementing 403-406
working 401, 402

randomized selection algorithm 378

range data type 10, 11

recursion 59, 60

recursive cases 59

regular binary tree 182

reverse Polish expression
parsing 196-200

reverse Polish notation (RPN) 196

root node 179, 180

running time complexity, algorithm
computing 52-54

S

searching algorithms 313, 314
- binary search 325-331
- exponential search 337-341
- interpolation search 331-337
- jump search 320-325
- linear search 314, 315
- selecting 341

search term 316

selection algorithms 377

selection by sorting 378

selection sort algorithm 356-359

separate chaining 272-277

sequence data types 9
- lists 11, 12
- range 10, 11
- string 9, 10

set 23, 24
- immutable sets 26
- operations 25
- Venn diagram 24

shortest path problem 76-89

siblings 181

simple binary tree 181

singly linked lists 98
- clearing 113
- creating 98
- creation, improving 99
- element, searching in 107
- intermediate node, deleting 111-113
- items, appending 100
- items, appending at intermediate positions 103-106
- items, appending to end of list 100-103
- items, deleting 108
- node, deleting at end 109-111
- node, deleting from beginning 108, 109
- querying 106
- size, obtaining 107, 108
- traversal, improving 99
- traversing 98

sink vertex 284

slicing operations 19

slot 252

sorting algorithms 345
- bubble sort algorithms 346-352
- insertion sort algorithm 352-356
- quicksort algorithm 359-364
- selection sort algorithm 356-359
- Timsort algorithm 369-373

source vertex 284

space complexity, algorithm 40, 41

stack-based queues 166
- approaches 166-169
- dequeue operation 170-173
- enqueue operation 170

stacks 141
- applications 155, 156
- example 142, 144
- implementing, with arrays 145-147
- implementing, with linked lists 148, 149
- operations 143
- peek operation 154
- pop operation 142, 143, 151, 153
- push operation 142, 143, 149-151

string matching algorithms 395

strings 9, 395
- + operator 10
- * operator 10
- prefix 396
- suffix 396

sublist **352**

substring **396**

subtree **180**

symbol tables **247, 278**
  example 278

T

theta notation **42-44**

time complexity, algorithm **38**
  average-case running time 40
  best-case running time 40
  constant amount of time 38
  running time 38, 39
  worst-case running time 40

Timsort algorithm **369-373**

trees **179**
  binary search tree (BST) 201, 202
  binary tree 181
  child node 181
  degree of node 180
  depth of node 181
  edge 181
  height 181
  leaf node 180
  level of root node 181
  node 180
  parent node 181
  root node 180
  siblings 181
  subtree 180

tree traversal **186**
  in-order tree traversal 186-188
  level-order traversal 191-193
  post-order tree traversal 190, 191
  pre-order tree traversal 188-190

tuples **18**
  operations 18

U

unbalanced binary tree **184**

undirected graph **283**

unordered linear search **315**
  implementation 316, 317

UserDict **32**

UserList **32**

UserString **33**

V

vertex **282**

W

weighted graph **285**

Windows operating system
  Python, installing for 2, 3

Z

zero-based indexing **19**