

Table of Contents

Preface	vii
Chapter 1: Why Bother? – Basic	1
The performance of an algorithm	2
Best case, worst case and the average case complexity	3
Analysis of asymptotic complexity	4
Asymptotic upper bound of a function	4
Asymptotic upper bound of an algorithm	8
Asymptotic lower bound of a function	8
Asymptotic tight bound of a function	9
Optimization of our algorithm	10
Fixing the problem with large powers	10
Improving time complexity	11
Summary	13
Chapter 2: Cogs and Pulleys – Building Blocks	15
Arrays	16
Insertion of elements in an array	16
Insertion of a new element and the process of appending it	18
Linked list	20
Appending at the end	21
Insertion at the beginning	23
Insertion at an arbitrary position	24
Looking up an arbitrary element	25
Removing an arbitrary element	27
Iteration	28
Doubly linked list	30
Insertion at the beginning or at the end	31
Insertion at an arbitrary location	32
Removing the first element	33
Removing an arbitrary element	34

Removal of the last element	36
Circular linked list	37
Insertion	38
Removal	38
Rotation	39
Summary	40
Chapter 3: Protocols – Abstract Data Types	41
Stack	42
Fixed-sized stack using an array	43
Variable-sized stack using a linked list	45
Queue	47
Fixed-sized queue using an array	48
Variable-sized queue using a linked list	50
Double ended queue	52
Fixed-length double ended queue using an array	53
Variable-sized double ended queue using a linked list	55
Summary	56
Chapter 4: Detour – Functional Programming	57
Recursive algorithms	58
Lambda expressions in Java	60
Functional interface	60
Implementing a functional interface with lambda	61
Functional data structures and monads	62
Functional linked lists	62
The forEach method for a linked list	65
Map for a linked list	66
Fold operation on a list	67
Filter operation for a linked list	70
Append on a linked list	74
The flatMap method on a linked list	74
The concept of a monad	76
Option monad	76
Try monad	82
Analysis of the complexity of a recursive algorithm	86
Performance of functional programming	89
Summary	90
Chapter 5: Efficient Searching – Binary Search and Sorting	91
Search algorithms	92
Binary search	93
Complexity of the binary search algorithm	96

Sorting	98
Selection sort	98
Complexity of the selection sort algorithm	101
Insertion sort	102
Complexity of insertion sort	104
Bubble sort	105
Inversions	106
Complexity of the bubble sort algorithm	107
A problem with recursive calls	108
Tail recursive functions	109
Non-tail single recursive functions	112
Summary	114
Chapter 6: Efficient Sorting – quicksort and mergesort	115
<hr/>	
quicksort	115
Complexity of quicksort	120
Random pivot selection in quicksort	122
mergesort	123
The complexity of mergesort	126
Avoiding the copying of tempArray	127
Complexity of any comparison-based sorting	129
The stability of a sorting algorithm	133
Summary	134
Chapter 7: Concepts of Tree	135
<hr/>	
A tree data structure	136
The traversal of a tree	139
The depth-first traversal	140
The breadth-first traversal	142
The tree abstract data type	143
Binary tree	145
Types of depth-first traversals	147
Non-recursive depth-first search	149
Summary	153
Chapter 8: More About Search – Search Trees and Hash Tables	155
<hr/>	
Binary search tree	156
Insertion in a binary search tree	158
Invariant of a binary search tree	160
Deletion of an element from a binary search tree	162
Complexity of the binary search tree operations	167
Self-balancing binary search tree	168
AVL tree	171
Complexity of search, insert, and delete in an AVL tree	176

Red-black tree	177
Insertion	179
Deletion	183
The worst case of a red-black tree	188
Hash tables	190
Insertion	191
The complexity of insertion	193
Search	193
Complexity of the search	194
Choice of load factor	194
Summary	194
Chapter 9: Advanced General Purpose Data Structures	195
Priority queue ADT	195
Heap	196
Insertion	197
Removal of minimum elements	198
Analysis of complexity	199
Serialized representation	200
Array-backed heap	201
Linked heap	204
Insertion	206
Removal of the minimal elements	207
Complexity of operations in ArrayHeap and LinkedHeap	209
Binomial forest	209
Why call it a binomial tree?	211
Number of nodes	212
The heap property	212
Binomial forest	214
Complexity of operations in a binomial forest	221
Sorting using a priority queue	221
In-place heap sort	222
Summary	223
Chapter 10: Concepts of Graph	225
What is a graph?	226
The graph ADT	228
Representation of a graph in memory	229
Adjacency matrix	230
Complexity of operations in a sparse adjacency matrix graph	234
More space-efficient adjacency-matrix-based graph	235
Complexity of operations in a dense adjacency-matrix-based graph	243

Adjacency list	244
Complexity of operations in an adjacency-list-based graph	250
Adjacency-list-based graph with dense storage for vertices	251
Complexity of the operations of an adjacency-list-based graph with dense storage for vertices	258
Traversal of a graph	259
Complexity of traversals	264
Cycle detection	264
Complexity of the cycle detection algorithm	267
Spanning tree and minimum spanning tree	267
For any tree with vertices V and edges E , $ V = E + 1$	268
Any connected undirected graph has a spanning tree	269
Any undirected connected graph with the property $ V = E + 1$ is a tree	269
Cut property	269
Minimum spanning tree is unique for a graph that has all the edges whose costs are different from one another	271
Finding the minimum spanning tree	272
Union find	273
Complexity of operations in UnionFind	277
Implementation of the minimum spanning tree algorithm	277
Complexity of the minimum spanning tree algorithm	279
Summary	280
Chapter 11: Reactive Programming	281
What is reactive programming?	282
Producer-consumer model	283
Semaphore	283
Compare and set	284
Volatile field	285
Thread-safe blocking queue	286
Producer-consumer implementation	289
Spinlock and busy wait	296
Functional way of reactive programming	301
Summary	313
Index	315
