

Contents

1	<i>Algorithms</i>	1
1.1	<i>Pretest</i>	2
1.2	<i>Algorithms and correctness</i>	5
	<i>Bounded linear search</i> 6 <i>Binary search</i> 15	
	<i>A linked list class</i> 24 <i>Selection sort</i> 29	
1.3	<i>Algorithms and efficiency</i>	35
	<i>A line-by-line analysis</i> 35	
	<i>Big-oh and big-theta defined formally</i> 44	
	<i>Constant factors</i> 50 <i>Recurrences</i> 54	
1.4	<i>Experiments</i>	57
	<i>Counting comparisons</i> 57 <i>Merge sort and quick sort</i> 65	
	<i>Measuring running time</i> 76	
	<i>Sorting algorithm summary</i> 80	
1.5	<i>Chapter summary</i>	84
2	<i>Data structures</i>	87
2.1	<i>Abstract data types</i>	88
	<i>The ADT canon</i> 89 <i>Lists</i> 90	
	<i>Stacks and queues</i> 92 <i>Sets</i> 96	
	<i>Maps</i> 97 <i>Bags</i> 100	
2.2	<i>Array-based data structures</i>	107
	<i>Random access</i> 108 <i>ArrayList</i> 109	
	<i>Efficiency of an array</i> 112 <i>ArrayStack</i> 114	
2.3	<i>Linked data structures</i>	116

	<i>LinkedStack</i>	118	<i>Recursion in the node</i>	120
2.4	<i>Data structures built from abstractions</i>			125
	<i>Multidimensional arrays</i>	125		
	<i>Ring buffers</i>	127	<i>Linked trees</i>	129
2.5	<i>Data structures adapted from ADTs</i>			138
	<i>Lists adapted as stack and queues</i>	138		
	<i>Lists adapted as maps</i>	141	<i>Maps adapted as bags</i>	143
	<i>Bags adapted as sets</i>	144		
2.6	<i>ADTs and data structures in other languages</i>			150
	<i>ADTs in C</i>	150	<i>ADTs in SML</i>	165
	<i>ADTs in Python</i>	173		
2.7	<i>Programming practices</i>			179
	<i>Assertions</i>	179	<i>Nested classes</i>	181
	<i>Iterators</i>	185	<i>Interfaces</i>	196
2.8	<i>The road ahead</i>			199
2.9	<i>Chapter summary</i>			203
3	<i>Case studies</i>			207
3.1	<i>Linear-time sorting algorithms</i>			208
	<i>Counting sort</i>	209	<i>Radix sort</i>	213
	<i>Bucket sort</i>	219		
3.2	<i>Disjoint sets and array forests</i>			223
	<i>The disjoint set ADT</i>	226	<i>The array forest data structure</i>	229
	<i>Find and union strategies</i>	232		
3.3	<i>Priority queues and heaps</i>			244
	<i>The priority queue ADT</i>	245		
	<i>Brute force implementations</i>	247		
	<i>The heap data structure</i>	253		
	<i>Heap sort</i>	262	<i>HeapPriorityQueue</i>	267
3.4	<i>N-Sets and bit vectors</i>			274
	<i>The N-set ADT</i>	277	<i>The bit vector data structure</i>	281
3.5	<i>Skip lists</i>			290
	<i>Map data structure performance</i>	290		
	<i>The skip list data structure</i>	294		
	<i>Skip list performance</i>	298		

3.6	<i>Chapter summary</i>	308
4	<i>Graphs</i>	311
4.1	<i>Concepts</i>	312
4.2	<i>Implementation</i>	317
	<i>Adjacency list and adjacency matrix</i>	320
	<i>Space and time efficiency</i>	326
	<i>Weighted graphs</i>	331
4.3	<i>Traversal</i>	334
	<i>Breadth-first and depth-first</i>	337
	<i>Implementing traversal algorithms</i>	342
	<i>The complexity of traversal</i>	348
	<i>Recursive depth-first traversal</i>	351
4.4	<i>Minimum spanning trees</i>	356
	<i>The MST problem</i>	357
	<i>Kruskal's algorithm</i>	362
	<i>Prim's algorithm</i>	370
	<i>Performance</i>	375
4.5	<i>Shortest paths</i>	379
	<i>The SSSP problem</i>	381
	<i>The Bellman-Ford algorithm</i>	387
	<i>Dijkstra's algorithm</i>	392
	<i>Routing algorithms</i>	397
4.6	<i>Chapter summary</i>	403
5	<i>Search trees</i>	407
5.1	<i>Binary search trees</i>	408
	<i>Iterative implementation</i>	413
	<i>Recursive implementation</i>	418
	<i>Object-oriented recursive implementation</i>	421
	<i>Performance</i>	426
5.2	<i>The balanced tree problem</i>	430
5.3	<i>AVL trees</i>	438
	<i>Violations</i>	440
	<i>Implementation</i>	445
	<i>Performance</i>	450
5.4	<i>Traditional red-black trees</i>	454
	<i>Insertion</i>	459
	<i>Deletion</i>	467
	<i>Performance</i>	475

5.5	<i>Left-leaning red-black trees</i>	478
	<i>Insertion</i> 481 <i>Deletion</i> 485	
	<i>Performance</i> 489	
5.6	<i>B-trees</i>	495
	<i>Two-three trees</i> 495 <i>Deriving B-trees</i> 504	
	<i>Implementing B-trees</i> 507 <i>Insertion</i> 513	
	<i>Performance</i> 522	
5.7	<i>Chapter summary</i>	525
6	<i>Dynamic programming</i>	529
6.1	<i>Overlapping subproblems</i>	530
	<i>Characterizing coin-changing</i> 535	
	<i>Computing the best bag of coins</i> 544	
6.2	<i>Three problems</i>	548
	<i>The knapsack problem</i> 548	
	<i>The longest common subsequence problem</i> 552	
	<i>The optimal matrix multiplication problem</i> 556	
6.3	<i>Elements of dynamic programming</i>	560
6.4	<i>Three solutions</i>	565
	<i>A solution to the knapsack problem</i> 565	
	<i>A solution to the longest common subsequence problem</i> 568	
	<i>A solution to the optimal matrix multiplication problem</i> 571	
6.5	<i>Optimal binary search trees</i>	581
	<i>The optimal BST problem</i> 583	
	<i>Building tables</i> 590	
6.6	<i>Natural-breaks classification</i>	595
	<i>Recursive formulation</i> 600	
	<i>A dynamic-programming algorithm</i> 602	
6.7	<i>Chapter summary</i>	606
7	<i>Hash tables</i>	609
7.1	<i>Hash table basics</i>	610
7.2	<i>Separate chaining</i>	619

	<i>Design</i>	619	<i>Implementation</i>	622
7.3	<i>Open addressing</i>	626		
	<i>Basic implementation</i>	629	<i>Deletion</i>	633
	<i>Alternate probe strategies</i>	640		
7.4	<i>Hash functions</i>	650		
	<i>Integer hash functions</i>	652		
	<i>String hash functions</i>	662		
7.5	<i>Perfect hashing</i>	668		
	<i>Universal hash functions</i>	669		
	<i>Design of the table</i>	673		
7.6	<i>Chapter summary</i>	677		
8	<i>String processing</i>	681		
8.1	<i>Sorting algorithms for strings</i>	686		
	<i>String quick sort</i>	688	<i>String bucket sort</i>	694
	<i>String radix sort</i>	697		
8.2	<i>Tries</i>	701		
	<i>Abstraction</i>	703	<i>Implementation</i>	706
8.3	<i>Huffman encoding</i>	715		
	<i>Text encoding schemes</i>	715		
	<i>Building the key</i>	723	<i>Optimality</i>	727
8.4	<i>Edit distance</i>	733		
8.5	<i>Grammars</i>	739		
	<i>Recursive descent parsing</i>	742		
	<i>CKY parsing described</i>	754		
	<i>CKY parsing implemented</i>	760		
8.6	<i>Chapter summary</i>	764		
	<i>Index</i>	775		