

Contents

<i>Preface</i>	<i>page ix</i>
1 Introduction	1
1.1 Functional vs. Imperative Data Structures	1
1.2 Strict vs. Lazy Evaluation	2
1.3 Terminology	3
1.4 Approach	4
1.5 Overview	4
2 Persistence	7
2.1 Lists	7
2.2 Binary Search Trees	11
2.3 Chapter Notes	15
3 Some Familiar Data Structures in a Functional Setting	17
3.1 Leftist Heaps	17
3.2 Binomial Heaps	20
3.3 Red-Black Trees	24
3.4 Chapter Notes	29
4 Lazy Evaluation	31
4.1 $\$$ -notation	31
4.2 Streams	34
4.3 Chapter Notes	37
5 Fundamentals of Amortization	39
5.1 Techniques of Amortized Analysis	39
5.2 Queues	42
5.3 Binomial Heaps	45
5.4 Splay Heaps	46
5.5 Pairing Heaps	52

5.6	The Bad News	54
5.7	Chapter Notes	55
6	Amortization and Persistence via Lazy Evaluation	57
6.1	Execution Traces and Logical Time	57
6.2	Reconciling Amortization and Persistence	58
6.2.1	The Role of Lazy Evaluation	59
6.2.2	A Framework for Analyzing Lazy Data Structures	59
6.3	The Banker's Method	61
6.3.1	Justifying the Banker's Method	62
6.3.2	Example: Queues	64
6.3.3	Debit Inheritance	67
6.4	The Physicist's Method	68
6.4.1	Example: Binomial Heaps	70
6.4.2	Example: Queues	72
6.4.3	Example: Bottom-Up Mergesort with Sharing	74
6.5	Lazy Pairing Heaps	79
6.6	Chapter Notes	81
7	Eliminating Amortization	83
7.1	Scheduling	84
7.2	Real-Time Queues	86
7.3	Binomial Heaps	89
7.4	Bottom-Up Mergesort with Sharing	94
7.5	Chapter Notes	97
8	Lazy Rebuilding	99
8.1	Batched Rebuilding	99
8.2	Global Rebuilding	101
8.2.1	Example: Hood–Melville Real-Time Queues	102
8.3	Lazy Rebuilding	104
8.4	Double-Ended Queues	106
8.4.1	Output-Restricted Deques	107
8.4.2	Banker's Deques	108
8.4.3	Real-Time Deques	111
8.5	Chapter Notes	113
9	Numerical Representations	115
9.1	Positional Number Systems	116
9.2	Binary Numbers	116
9.2.1	Binary Random-Access Lists	119
9.2.2	Zeroless Representations	122

9.2.3	Lazy Representations	125
9.2.4	Segmented Representations	127
9.3	Skew Binary Numbers	130
9.3.1	Skew Binary Random-Access Lists	132
9.3.2	Skew Binomial Heaps	134
9.4	Trinary and Quaternary Numbers	138
9.5	Chapter Notes	140
10	Data-Structural Bootstrapping	141
10.1	Structural Decomposition	142
10.1.1	Non-Uniform Recursion and Standard ML	143
10.1.2	Binary Random-Access Lists Revisited	144
10.1.3	Bootstrapped Queues	146
10.2	Structural Abstraction	151
10.2.1	Lists With Efficient Catenation	153
10.2.2	Heaps With Efficient Merging	158
10.3	Bootstrapping To Aggregate Types	163
10.3.1	Tries	163
10.3.2	Generalized Tries	166
10.4	Chapter Notes	169
11	Implicit Recursive Slowdown	171
11.1	Queues and Deques	171
11.2	Catenable Double-Ended Queues	175
11.3	Chapter Notes	184
A	Haskell Source Code	185
	<i>Bibliography</i>	207
	<i>Index</i>	217

