

Enno Ohlebusch

Bioinformatics Algorithms

Sequence Analysis, Genome Rearrangements,
and Phylogenetic Reconstruction

Contents

Preface	xiii
1 Molecular Biology in a Nutshell	1
1.1 Nucleic acids and proteins	1
1.2 Evolution	7
2 Exact String Matching	9
2.1 Basic string definitions	9
2.2 The naive algorithm	11
2.3 The Boyer-Moore-Horspool algorithm	12
2.4 The Knuth-Morris-Pratt algorithm	15
2.5 The Aho-Corasick algorithm for a set of patterns	24
3 Answering Range Minimum Queries in Constant Time	33
3.1 Basic definitions	33
3.2 Range minimum vs. lowest common ancestor	34
3.3 Range minimum queries	42
3.3.1 The sparse table algorithm	42
3.3.2 An optimal algorithm	43
3.4 Completing the proof of correctness	53
4 Enhanced Suffix Arrays	59
4.1 Suffix arrays	59
4.1.1 Linear-time construction	61
4.1.2 Induced sorting	68
4.2 The LCP-array	79
4.2.1 Linear-time construction	79
4.2.2 Longest common prefix	84
4.3 The lcp-interval tree	85
4.3.1 Finding child and parent intervals	88
4.3.2 Bottom-up traversal	93
4.3.3 Top-down traversal	98

4.3.4	Finding child intervals without RMQs	105
4.4	Suffix trees	110
4.4.1	Linear-time construction	113
5	Applications of Enhanced Suffix Arrays	115
5.1	Exact string matching	116
5.1.1	Forward search on suffix trees	116
5.1.2	Forward search on suffix arrays	117
5.1.3	Binary search	120
5.2	Lempel-Ziv factorization	125
5.2.1	Longest previous substring	126
5.2.2	Ultra-fast factorization	134
5.3	Finding repeats	138
5.3.1	Longest repeats	140
5.3.2	Supermaximal repeats	144
5.3.3	Maximal repeats	148
5.3.4	Maximal repeated pairs	149
5.3.5	Non-overlapping repeats	155
5.3.6	Maximal periodicities	157
5.4	Comparing two strings	173
5.4.1	Generalized suffix array	173
5.4.2	Longest common substring	181
5.4.3	Finding exact matches	183
5.5	Traversals with suffix links	185
5.5.1	Suffix links in the suffix tree	185
5.5.2	Suffix links in the lcp-interval tree	186
5.5.3	Computing suffix links space efficiently	187
5.5.4	Matching statistics	194
5.5.5	Merging two suffix arrays in linear time	203
5.6	Comparing multiple strings	206
5.6.1	Generalized suffix array	206
5.6.2	Longest common substring	208
5.6.3	Document frequency	216
5.6.4	Document retrieval	221
5.6.5	Shortest unique substrings	224
5.6.6	A distance measure for genomes	228
5.6.7	All-pairs suffix-prefix matching	231
5.7	String kernels	237
5.7.1	Machine learning	237
5.7.2	Calculating a string kernel	238
5.7.3	Calculating the kernel matrix	243
5.7.4	Classification	243
5.7.5	The TF-IDF weighting scheme	244

5.8	String mining	247
	5.8.1 Extraction phase	248
	5.8.2 Intersection phase	250
6	Making the Components of Enhanced Suffix Arrays Smaller	257
6.1	Constant time <i>rank</i> and <i>select</i> queries	257
6.2	Compressed suffix and LCP-arrays	262
	6.2.1 Compressed suffix array	262
	6.2.2 Compressed LCP-array	264
6.3	The balanced parentheses sequence of the LCP-array	265
	6.3.1 Finding the parent interval	270
	6.3.2 Finding child intervals	272
	6.3.3 Computing $getInterval([i..j], c)$	274
	6.3.4 Answering RMQs in constant time	275
	6.3.5 Computing suffix link intervals	276
	6.3.6 Attaching additional information	276
7	Compressed Full-Text Indexes	281
7.1	The components of a compressed full-text index	281
7.2	The Burrows-Wheeler transform	282
	7.2.1 Encoding	282
	7.2.2 Decoding	284
	7.2.3 Data compression	287
	7.2.4 Direct construction of the BWT	291
7.3	Backward search	299
	7.3.1 A simple FM-index	299
	7.3.2 The search algorithm	300
7.4	Wavelet trees	303
	7.4.1 Answering <i>rank</i> and <i>select</i> queries	304
	7.4.2 Retrieval of $SA[i]$ and the string starting at $SA[i]$	306
	7.4.3 Implementation: If σ is a power of 2	307
	7.4.4 Implementation: If σ is not a power of 2	310
	7.4.5 Other types of wavelet trees	315
7.5	Analyzing a string space efficiently	315
	7.5.1 Construction of the LCP-array from the BWT	315
	7.5.2 Bottom-up traversal of the lcp-interval tree	321
	7.5.3 Shortest unique substrings	322
	7.5.4 Top-down traversal of the lcp-interval tree	323
	7.5.5 Finding repeats	328
	7.5.6 Lempel-Ziv factorization	332
7.6	Space-efficient comparison of two strings	336
	7.6.1 Matching statistics	336
	7.6.2 Maximal exact matches	340
	7.6.3 Merging Burrows-Wheeler transformed strings	342

7.7	Space-efficient comparison of multiple strings	345
7.7.1	Document array, LCP-array, and correction terms	345
7.7.2	Document retrieval with wavelet trees	348
7.7.3	All-pairs suffix-prefix matching	352
7.8	Bidirectional search	357
7.8.1	Burrows-Wheeler transform of the reverse string	358
7.8.2	The suffix array of the reverse string	364
7.8.3	The lcp-array of the reverse string	366
7.8.4	The bidirectional search algorithm	369
7.9	Approximate string matching	374
7.9.1	Using backward search	375
7.9.2	Using bidirectional search	380
8	Sequence Alignment	385
8.1	Pairwise alignment	386
8.1.1	Distance methods	387
8.1.2	Computing an optimal alignment in linear space	393
8.1.3	Edit distance	398
8.1.4	Similarity methods	399
8.1.5	Distance vs. similarity	401
8.1.6	General similarity functions and gap penalties	402
8.2	Multiple alignment	406
8.2.1	Pruning the search space	409
8.2.2	A 2-approximation algorithm	411
8.2.3	Progressive alignment	415
8.3	Whole genome alignment	417
8.3.1	Basic definitions and concepts	418
8.3.2	A global chaining algorithm	420
8.3.3	Alternative data structures	423
8.3.4	Longest/heaviest increasing subsequence	424
9	Sorting by Reversals	429
9.1	Introduction	429
9.2	Basic definitions	437
9.3	The reality-desire diagram	440
9.4	Components	445
9.4.1	Elementary intervals	445
9.4.2	Finding cycles and components	448
9.5	Sorting a permutation without bad components	451
9.6	Dealing with bad components	455
9.6.1	Hurdles	458
9.6.2	A fortress	461
9.7	Sorting by reversals in quadratic time	467
9.7.1	Finding a happy clique	468

9.7.2 Searching the happy clique	473
10 Phylogenetic Reconstruction	477
10.1 Introduction	477
10.1.1 Methods of phylogenetic inference	479
10.1.2 Molecular anthropology	481
10.2 Basic definitions	489
10.3 Ultrametric distance matrices and trees	492
10.3.1 Characterization of ultrametric matrices	494
10.3.2 Construction algorithm	497
10.3.3 The UPGMA-algorithm	501
10.3.4 Fast UPGMA implementation based on quadrees	509
10.4 Additive distance matrices and trees	514
10.4.1 Ultrametric trees revisited	515
10.4.2 Reduction of the additive tree problem	516
10.4.3 Characterization of additive matrices	521
10.4.4 Construction algorithm	524
10.4.5 Splits and quartets	526
10.4.6 Uniqueness of the additive tree	528
10.5 Neighbor-joining algorithms	531
10.5.1 Farris' neighbor-joining algorithm	534
10.5.2 Saitou and Nei's neighbor-joining algorithm	540
10.5.3 Fast neighbor-joining	546
10.6 Non-additive dissimilarity matrices	548
10.6.1 Nearly additive matrices and quartet-consistency .	549
10.6.2 Estimating edge weights	561
10.6.3 Bootstrapping	569
Bibliography	571
Index	599