

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Improving Current DR Solutions . . . . .	4
1.2.1	Design of Data Structures to Use in DR Solutions . . . . .	4
1.2.2	Design of Compressed Indexes to solve DR Problems . . . . .	4
1.3	Thesis Statement . . . . .	5
1.3.1	Thesis Contributions . . . . .	5
1.4	Thesis Structure . . . . .	8
<b>2</b>	<b>Basic Background</b>	<b>9</b>
2.1	Text Compression . . . . .	9
2.2	Measures of Compressibility . . . . .	11
2.3	Bitvectors . . . . .	13
2.4	Fundamental Text Indexes . . . . .	13
2.5	Other Useful Data Structures . . . . .	15
2.5.1	Wavelet Trees . . . . .	15
2.5.2	Succinct Tree Representations . . . . .	17
2.5.3	Cartesian Tree . . . . .	18
2.5.4	Lowest Common Ancestor and Range Minimum Query . . . . .	18
2.6	Compressed Text Indexes Based on the SA . . . . .	19
2.6.1	The Compressed Suffix Array . . . . .	20
2.6.2	The FMI Family . . . . .	21
2.6.3	The Locally Compressed Suffix Array . . . . .	22
2.7	LZ-Based Compressors . . . . .	24
2.7.1	LZ77 Compression . . . . .	24
2.7.2	LZ78 Compression . . . . .	25
2.8	The LZ-Index . . . . .	26
2.8.1	The Basic Structure . . . . .	26
<b>3</b>	<b>Document Retrieval Review</b>	<b>29</b>
3.1	Document Listing . . . . .	29
3.2	Top- $k$ Retrieval . . . . .	32
3.3	Document Listing in Repetitive Texts . . . . .	34
<b>4</b>	<b>Contributions in Text Indexing</b>	<b>38</b>
4.1	Structures for Compressed Suffix Arrays . . . . .	38

4.1.1	Elias-Fano Coding . . . . .	39
4.1.2	The Suffix Array of Grossi and Vitter . . . . .	39
4.1.3	The Suffix Array of Rao . . . . .	42
4.1.4	Experimental Results . . . . .	43
4.2	Hybrid Indexing on Repetitive Datasets . . . . .	52
4.2.1	Hybrid Indexing . . . . .	52
4.2.2	Implementation . . . . .	54
4.3	Experiments . . . . .	55
4.4	Conclusions . . . . .	58
<b>5</b>	<b>Improved Range Minimum Queries</b>	<b>59</b>
5.1	State of the Art . . . . .	60
5.2	A Simplified Implementation . . . . .	64
5.2.1	Construction . . . . .	65
5.3	Implementing Balanced Parentheses . . . . .	66
5.4	Experimental Results . . . . .	67
5.5	Conclusions . . . . .	69
<b>6</b>	<b>An LZ-based Index for Document Listing</b>	<b>71</b>
6.1	Structure . . . . .	72
6.2	Queries . . . . .	74
6.3	Implementation . . . . .	76
6.3.1	Experimental Results . . . . .	77
6.4	Conclusions . . . . .	83
<b>7</b>	<b>An LZ-based Index for Top-k Retrieval</b>	<b>85</b>
7.1	Structure . . . . .	86
7.2	Queries . . . . .	86
7.3	Improving the Quality . . . . .	88
7.4	Experimental Results . . . . .	89
7.4.1	Space study . . . . .	89
7.4.2	Space/time tradeoffs . . . . .	89
7.4.3	Quality . . . . .	93
7.5	Conclusions . . . . .	96
<b>8</b>	<b>An LZ77 Based Index for Document Listing</b>	<b>98</b>
8.1	A Document Listing Approach Based on the Hybrid-Index . . . . .	98
8.1.1	The Structure for Primary Matches . . . . .	99
8.1.2	The Structure for Secondary Matches . . . . .	99
8.1.3	The Document Listing Algorithm . . . . .	104
8.1.4	Reducing the Size of the Inverted Lists . . . . .	105
8.2	Including Frequencies . . . . .	106
8.3	Conclusions . . . . .	107
<b>9</b>	<b>Conclusions and Further Research</b>	<b>108</b>
	<b>References</b>	<b>111</b>

# List of Tables

2.1	Time-space complexities of most popular CSAs. The time construction for these indexes is $O(n \log \sigma)$ . . . . .	21
2.2	Time-space complexity of main self-indexes of the FM-Index family. The time construction for these indexes is generally $O(n \log \sigma)$ . . . . .	22
4.1	Main characteristics for the texts considered in the experiments with the indexes. We show the entropy of order 0 and 4, and also the real compressibility for these texts, using the best-known compressors: <code>gzip</code> (option <code>-9</code> ), <code>bzip2</code> (option <code>-9</code> ) and <code>PPMDi</code> (option <code>-1 9</code> ). . . . .	43
4.2	Sizes in MB of the uncompressed files, the files compressed with <code>7zip</code> and the three indexes: the LZ77-Index of Krefl and Navarro (with default values), the FM-Indexes with SA-FMI-sampling in 32 and 256, and the hybrid indexes with maximum patterns lengths $M$ in 50 and 100, with SA-FMI-sampling in 32 and 256 in the internal FM-Index for the filtered text. Between parentheses are the parameter values for the FMIs and the hybrid indexes. . . . .	56
6.1	Main characteristics of the text collections. . . . .	78
6.2	Space breakdown of the main components in our LZ-DLIndex structure, with values in bpc. For RevTrie and Doc columns the space is the sum of the components detailed below them (bpc values in italics). The Range columns does not include the RMQ structures to speed up the index. The percentages refer to the total size of the index. The column $(/ \text{LZ78} )$ indicates the ratio of the total size over $ \text{LZ78} $ , and the last column, in turn, gives also $(n/n')$ . . . . .	79
6.3	Number of occurrences of each type, for pattern lengths $m = 6$ and $m = 10$ . Under each number, we give the percentages of the documents output. For the three types of occurrences these refer to <code>ndoc</code> , and for column <code>ndoc</code> this refers to $D$ . . . . .	80
8.1	New points in the grid $G$ for the three phrases of our example at Figure 8.2. . . . .	102

# List of Figures

2.1	A <i>Trie</i> on the strings: $she_{\$1}$ , $sees_{\$2}$ and $cheese_{\$3}$ , and also the GST and GSA for the text: $she_{\$1}sees_{\$2}cheese_{\$3}$ . . . . .	14
2.2	The <i>Wavelet Tree</i> for the sequence $S="she\_sees\_cheese"$ over the alphabet $\Sigma = \{\$, \_, c, e, h, s\}$ . . . . .	15
2.3	A wavelet tree showing the nodes that cover a leaves range. . . . .	16
2.4	A <i>Cartesian Tree</i> on a input array . . . . .	18
2.5	The <i>2d-Min-Heap</i> data structure on the array $A$ and the BPS, DFUDS and LOUDS succinct representations of the tree. . . . .	20
2.6	The resulting phrases after applying the LZ77 parsing on a text and the two types of occurrences in the parsed text . . . . .	24
2.7	The resulting phrases when applying the LZ78 parsing of a collection with 3 texts. . . . .	25
2.8	The three types of occurrences according to how they span blocks (or phrases). . . . .	26
2.9	The structures to report occurrences of type 1. . . . .	26
2.10	The scheme to report the occurrences of type 2. . . . .	27
3.1	The generalized suffix tree a text and the arrays $E$ and $C$ that form the structure of Muthukrisman to solve DL queries. . . . .	30
4.1	GVCSA with $t = 2$ levels of decomposition for a suffix array. . . . .	40
4.2	RaoCSA with $t = 1$ level of decomposition and for $l = 4$ for a suffix array. . . . .	43
4.3	Space/time tradeoffs for accessing one cell using various options for $(t, l)$ for GVCSA. . . . .	45
4.4	Various options for $(t, l)$ for RaoCSA. On the left we show the basic scheme, on the right our improvement using wavelet trees. . . . .	46
4.5	Various options for $(t, l)$ for our improvement in RaoCSA using runs and wavelet trees. . . . .	47
4.6	Time-space tradeoffs to access one cell. On the left, basic GVCSA versus the version with runs, for all the texts. On the right, the best variants of RaoCSA. . . . .	48
4.7	Construction time and space for the different indexes on each text. . . . .	50
4.8	Time/space tradeoffs to access one random cell for the different indexes on each text. . . . .	51
4.9	The Basic scheme to find secondary matches. . . . .	54
4.10	Index sizes for prefixes of <i>cere</i> of 100, 200, 300 and 400 MB. . . . .	56
4.11	Average query times for the different indexes to locate occurrences with patterns of different lengths. . . . .	57

4.12	Index sizes and <code>locate</code> query time for the Hybrid-Index against the LZ77-Index.	58
5.1	An example array (top right) and its Cartesian tree (left).	61
5.2	The same arrangement of Figure 5.1, now on the DFUDS representation of the Cartesian tree.	63
5.3	The general tree (at middle) derived from the example Cartesian tree.	64
5.4	Query space and time on random arrays, for ranges of size 10,000, comparing the standard with our new implementations.	68
5.5	Query space and time on random arrays, for ranges of size 10,000.	68
5.6	Query time on random arrays, for ranges of increasing size and two values of $n$ .	69
5.7	Query time on pseudo-sorted arrays, $n = 10^6$ and ranges of size 10,000.	70
6.1	The structures to report documents for occurrences of type 1.	73
6.2	The scheme to report the occurrences of type 2 using RMQ structures in each level of the wavelet tree of Range.	76
6.3	Space/time comparison for pattern length $m = 6$ .	81
6.4	Space/time comparison for pattern length $m = 10$ .	82
6.5	Fraction of the real answer of our LZ-DLIndex for real queries, as a function of the prefix size of <code>TodoCLin</code> GB, for words and phrases of two words.	84
7.1	The main data structures of our approximate top- $k$ index.	88
7.2	Space breakdown of our structures for different $g$ values ( $g$ is the x-axis).	90
7.3	Space/time comparison for pattern length $m = 6$ (left) and $m = 10$ (right). Space (bpc) is the x-axis.	91
7.4	Space/time comparison for pattern length $m = 6$ (left) and $m = 10$ (right). Space (bpc) is the x-axis.	92
7.5	Recall of our approximate top- $k$ solution, as a function of the fraction of the answer (x-axis).	94
7.6	Quality of our approximate top- $k$ solution, as a function of the pattern length, for top-10 (left) and top-100 (right).	95
7.7	Fraction of the real answer found by LZ-AppTopK for real queries, as a function of the prefix size of <code>TodoCL</code> for words (left) and phrases of two words (right).	95
7.8	Fraction of the real answer found by LZ-TopkApp as a function of the prefix size of <code>TREC</code> , for arbitrary patterns of lengths 6 and 10, in top-10 and top-100.	96
8.1	The basic scheme with non-overlapped phrase sources.	100
8.2	An example with non-overlapped phrase sources distributed in two documents.	101
8.3	An example with several overlapped phrase sources in a document $d_U$ .	103