A. Apostolico M. Crochemore Z. Galil
U. Manber (Eds.)

Combinatorial Pattern Matching

Third Annual Symposium
Tucson, Arizona, USA, April 29-May 1, 1992
Proceedings

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Volume Editors

Alberto Apostolico
Dept. of Computer Sciences, Purdue University
1398 Comp. Sc. Bldg., West Lafayette, IN 47907-1398, USA

Maxime Crochemore
L. I. T. P., Université Paris VII
F-75251 Paris CX 05, France

Zvi Galil
Columbia University, New York, NY 10027, USA
and
Tel Aviv University
Ramat Aviv, Tel Aviv, Israel

Udi Manber
Computer Science Dept., University of Arizona
Gould-Simpson 721, Tucson, AZ 85721, USA

CR Subject Classification (1991): F.2.2, I.5.4, I.5.0, I.7.3, H.3.3, E.4

ISBN 3-540-56024-6 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-56024-6 Springer-Verlag New York Berlin Heidelberg

Typesetting: Camera ready by author/editor
45/3140-543210 - Printed on acid-free paper

Foreword

The papers contained in this volume were presented at the third annual symposium on Combinatorial Pattern Matching, held April 29 to May 1, 1992 in Tucson, Arizona. They were selected from 39 abstracts submitted in response to the call for papers.

Combinatorial Pattern Matching addresses issues of searching and matching of strings and more complicated patterns such as trees, regular expressions, extended expressions, etc. The goal is to derive nontrivial combinatorial properties for such structures and then to exploit these properties in order to achieve superior performances for the corresponding computational problems. In recent years, a steady flow of high-quality scientific study of this subject has changed a sparse set of isolated results into a full-fledged area of algorithmics. Still, there is currently no central place for disseminating results in this area. We hope that CPM can grow to serve as the focus point.

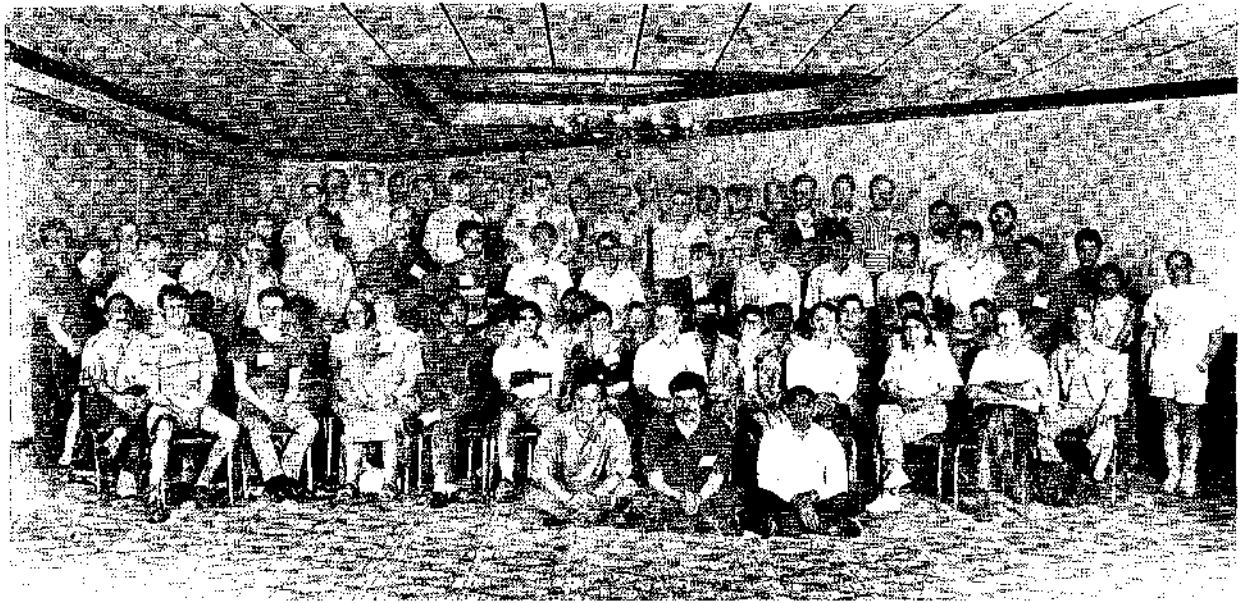
This area is expected to grow even further due to the increasing demand for speed and efficiency that comes especially from molecular biology and the Genome project, but also from other diverse areas such as information retrieval (e.g., supporting complicated search queries), pattern recognition (e.g., using strings to represent polygons and string matching to identify them), compilers (e.g., using tree matching), data compression, and program analysis (e.g., program integration efforts). The stated objective of CPM gatherings is to bring together once a year the researchers active in the area for an informal and yet intensive exchange of information about current and future research in the area.

The first two meetings were held at the University of Paris in 1990 and at the University of London in 1991. These two meetings were informal and no proceedings were produced. We hope that these proceedings will contribute to the success and growth of this area.

The conference was supported in part by the National Science Foundation and the University of Arizona.

Program Committee

A. Apostolico
M. Crochemore
Z. Galil
G. Gonnet
D. Gusfield
D. Hirschberg
U. Manber, *chair*
E. W. Myers
F. Tompa
E. Ukkonen



Standing back: (right to left): Gary Benson, Ricardo Baeza-Yates, Andrew Hume, Jim Knight, Christian Burks, Hal Berghel, Andrzej Ehrenfeucht, David Roach, Udi Manber, Mike Waterman, Ramana Idury, Amihood Amir, Gene Lawler, Ethan Port, William Chang, Martin Vingron, Pavel Pevzner, Esko Ukkonen, Frank Olken, Xin Xu, Alberto Apostolico, Thierry Lecroq, and George Havas. **Standing First Row:** Boris Pittel, Mudita Jain, Dominique Revuz, Gene Myers, Alejandro Schaffer, Steve Seiden, Dinesh Mehta, Tariq Choudhary, Tak Cheung Ip, L.J. Cummings, Rob Irving, Sampath Kannan, John Kececioglu, Pekka Kilpelainen, Kaizhong Zhang, Sun Wu, Lucas Hui, Tandy Warnow, and Yuh Dauh Lyuu. **Sitting second row:** Robert Paige, Ladan Rostami, Jong Yong Kim, Martin Farach, Haim Wolfson, Gad Landau, Jeanette Schmid, Glen Herrmannsfeldt, David Sankoff, Ramesh Hariharan, Laura Toniolo, Cari Soderlund, Dan Gusfield, and Wojciech Szpankowski. **Sitting first row:** Maxime Crochemore, Deborah Joseph, Xiaoqui Huang, Mereille Regnier, Dan Hirschberg, Marcella McClure, Gary Lewandowski, Taha Vasi, Brenda Baker, Campbell Fraser, and Philippe Jacquet. **On the floor:** John Oommen, Guy Jacobson, and Kiem Phong Vo.

Table of Contents

Probabilistic Analysis of Generalized Suffix Trees

(Extended Abstract)

Wojciech Szpankowski

Department of Computer Science, Purdue University,
W. Lafayette, IN 47907, U.S.A.

Abstract. Suffix trees find several applications in computer science and telecommunications, most notably in algorithms on strings, data compressions and codes. We consider in a probabilistic framework a family of generalized suffix trees – called b-suffix trees – built from the first n suffixes of a random word. In this family of trees, a noncompact suffix trees (i.e., such that every edge is labeled by a single symbol) is represented by $b = 1$, and a compact suffix tree (i.e., without unary nodes) is asymptotically equivalent to $b \to \infty$. Several parameters of b-suffix trees are of interest, namely the typical depth, the depth of insertion, the height, the external path length, and so forth. We establish some results concerning typical, that is, *almost sure* (a.s.), behavior of these parameters. These findings are used to obtain several insights into certain algorithms on words and universal data compression schemes.

1. Introduction

In recent years there has been a resurgence of interest in algorithmic and combinatorial problems on words due to a number of novel applications in computer science, telecommunications, and most notably in molecular biology. In computer science, several algorithms depend on a solution to the following problem: given a word X and a set of $b+1$ arbitrary suffixes $S_1, \ldots, S_{b+1}$ of X, what is the longest common prefix of these suffixes (cf. [2], [6], [8], [15], [16], [27], [28]). In coding theory (e.g., prefix codes) one asks for the shortest prefix of a suffix S_i which is not a prefix of any other suffixes S_j, $1 \le j \le n$ of a given sequence X. In data compression schemes, the following problem is of prime interest: for a given "data base" subsequence of length n, find the longest prefix of the $n+1$st suffix S_{n+1} which is not a prefix of any other suffixes S_i $(1 \le i \le n)$ of the underlying sequence X (cf. [21], [29], [17]). And last, but not least, in comparing molecular sequences (e.g., finding homology between DNA sequences) one may search for the longest run of a given motif (pattern) (cf. [11]). These, and several other problems on words, can be efficiently solved and analyzed by a clever manipulation of a data structure known as *suffix tree* [2], [19], [27]).

In general, a suffix tree is a digital tree built from suffixes of a given word X, and therefore it fits into the class of digital search indexes ([14]). A digital tree stores n

* This research was supported in part by NSF Grants CCR-8900305 and INT-8912631, and AFOSR Grant 90-0107, NATO Grant 0057/89, and Grant R01 LM05118 from the National Library of Medicine.

strings $\{S_1, \ldots, S_n\}$ built over a finite alphabet Σ. If the strings $\{S_1, \ldots, S_n\}$ are statistically independent and every edge is labelled by a single symbol from Σ, then the resulting digital tree is called a regular (or independent) trie ([1], [9], [14]). If all unary nodes of a trie are eliminated, then the tree becomes the PATRICIA trie (cf. [9], [14], [23]). Finally, if an external node in a regular trie can store up to b strings (keys), then such a tree is called a b-tric. As mentioned above, a suffix tree is a special tric in which the strings $\{S_1, \ldots, S_n\}$ are suffixes of a given sequence X. Note that in this case the strings are statistically dependent!

As in the case of regular tries, there are several modifications of the standard suffix tree. In a *noncompact suffix tree* – called also spread suffix tree and position tree – each edge is labelled by a letter from the alphabet Σ. If all unary nodes are eliminated in the noncompact version of the suffix tree, then the resulting tree is called a *compact suffix tree* (cf. [2]). Gonnet and Baeza-Yates [9] coined a name PAT for such a suffix tree to resemble the name PATRICIA used for compact tries. Here, we also adopt this name. In addition, we introduce a family of suffix trees – called b-suffix trees – parametrized by an integer $b \geq 1$. A tree in such a family is constructed from the noncompact suffix tree by eliminating all unary nodes b levels above the fringe (bottom level) of the tree (later we slightly modify this definition). These trees have several useful applications in algorithms on words, data compressions, and so forth, but more importantly b-suffix trees form a spectrum of trees with noncompact suffix trees ($b = 1$) at one extreme and compact suffix trees ($b \to \infty$) at the other extreme. This allows to assess some properties of PAT trees in a unified and substantially easier manner (cf. [23]).

In this extended abstract, we offer a characterization of generalized suffix trees in a probabilistic framework. (Most of the proofs are omitted, and can be found in the extended version [26].) Our probabilistic model is a very general one, namely we allow symbols of a string to be dependent. Moreover, instead of concentrating on a specific algorithm we present a list of results concerning several parameters of suffix trees, namely: the typical depth $D_n^{(b)}$, depth of insertion $L_n^{(b)}$, height $H_n^{(b)}$ and the shortest feasible path $s_n^{(b)}$. For example, the typical depth D_n^{PAT} for the PAT tree built from the string $P\$T$ where P and T are the pattern and the text strings respectively, is used by Chang and Lawler [6] in their design of an approximate string matching algorithm. On the other hand, the depth of insertion $L_n^{(1)}$ of a noncompact suffix tree is of prime interest to the complexity of the Lempel-Ziv universal compression scheme (cf. [24]), and $L_n^{(1)}$ is responsible for a *dynamic* behavior of many algorithms on words. Furthermore, the height and the shortest feasible path path indicate how balanced a typical suffix tree is, that is, how much one has to worry about worst-case situations.

Our main results can be summarized as follows. For a b-suffix tree built over an *unbounded word* X, we prove that the normalized height $H_n^{(b)}/\log n$, the normalized shortest feasible path $s_n^{(b)}/\log n$ and the normalized depth $D_n^{(b)}/\log n$ almost surely (a.s.) converge to $1/h_2^{(b)}$, $1/h_1$ and $1/h$ respectively, where for every $1 \leq b \leq \infty$ we have $h_2^{(b)} < h < h_1$. In the above, h is the entropy on the alphabet Σ, while the parameters h_1 and $h_2^{(b)}$ depend of the underlying probabilistic model. If the word has *finite* length, then the above results also hold except for the shortest feasible path

which clearly becomes $s_n^{(b)} = 1$ (see Remark 2(iii)). The most interesting behavior reveals the depth of insertion $L_n^{(b)}$ which converges *in probability* (pr.) to $(1/h)\log n$ but not almost surely. We prove that almost surely $L_n^{(b)}/\log n$ oscillates between $1/h_1$ and $1/h_2^{(b)}$. More interestingly, almost sure behavior of the *compact suffix tree* (i.e., PAT tree) can be deduced from the appropriate asymptotics of the b-suffix trees by taking $b \to \infty$. It is worth mentioning that all these results are obtained in a uniform manner by a technique that encompasses the so called string-ruler approach (cf. [13], [20]) and the mixing condition technique. Finally, using our results, we establish the average complexity of some exact and approximate pattern matching algorithms such as Chung-Lawler [6] and others (cf. [6]), etc. In addition, the results for noncompact suffix trees (cf. [25]) were used by us to settle in the negative the conjecture of Wyner and Ziv [29] concerning the length of the repeated pattern in a universal compression scheme (cf. [24]). In this paper, we prove the results already announced in [25] concerning the length of the last block in the Lempel-Ziv parsing algorithm [17].

Asymptotic analyses of suffix trees are very scanty in literature, and most of them deal with *noncompact suffix trees*. To the best of our knowledge, there are no probabilistic results on b-suffix trees and *compact suffix trees*. This can be easily verified by checking Section 7.2 of Gonnet and Baeza-Yates' book [9] which provides an up-to-date compendium of results concerning data structures and algorithms. The average case analysis of noncompact suffix trees was initialized by Apostolico and Szpankowski [3]. For the Bernoulli model (independent sequence of letters from a finite alphabet) the asymptotic behavior of the height was recently obtained by Devroye *et al.* [7], and the limiting distribution of the typical depth in a suffix tree is reported in Jacquet and Szpankowski [13]. Recently, Szpankowski [25] extended these results to a more general probabilistic model for *noncompact* suffix trees, that is, with $b = 1$. Finally, heuristic arguments were used by Blumer *et al.* [5] to show that the average number of internal nodes in a suffix tree is a linear function of n, and a rigorous proof of this can be found in [13]. Some related topics were discussed by Guibas and Odlyzko in [11].

2. Main Results and Their Consequences

In this paper, we consider a family of suffix trees called b-suffix trees. A tree in such a family has no unary nodes in all b levels above the fringe level of the corresponding noncompact suffix tree. Note that noncompact and compact suffix trees lie on two extremes of the spectrum of b-suffix trees; namely, a 1-suffix tree is a noncompact suffix tree, and a b-suffix tree becomes a compact suffix tree when $b \to \infty$. For the purpose of our analysis, however, a modified definition of b-suffix trees is more convenient. Hereafter, by b-suffix tree we mean a suffix tree *built from n first suffixes of an unbounded sequence $X = \{X_k\}_{k=1}^{\infty}$ that can store up to b suffixes in an external node*. We denote such a suffix tree by $\mathcal{S}_n^{(b)}$.

In this paper, we analyze six parameters of b-suffix trees $\mathcal{S}_n^{(b)}$; namely, the mth *depth* $L_n^{(b)}(m)$, the *height* $H_n^{(b)}$ and the *shortest feasible path* $s_n^{(b)}$, the *typical depth* $D_n^{(b)}$, the *depth of insertion* $L_n^{(b)}$ and the *external path length* $E_n^{(b)}$. The depth of the mth suffix is equal to the number of internal nodes in a path from the root to the