C. 01 -684

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

Combinatorial Pattern Matching

4th Annual Symposium, CPM 93 Padova, Italy, June 2-4, 1993 Proceedings



Springer-Verlag

Berlin Heidelberg New York London Paris Tokyo Hong Kong Barcelona Budapest Series Editors

Gerhard Goos Universität Karlsruhe Postfach 6980 Vincenz-Priessnitz-Straße 1 W-7500 Karlsruhe, FRG Suris Hartmanis Cornell University Department of Computer Science 4130 Upson Hall Ithaca, NY 14853, USA

Volume Editors

Alberto Apostolico Computer Science Dept., Purdue University West Lafayette, IN 47907-1398, USA and Dipartimento di Elettronica e Informatica, Università di Padova 1-35131 Padova, Italy

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, G.2.1, J.3

ISBN 3-540-56764-X Springer-Verlag Berlin Heidelberg New York ISBN 0-387-56764-X Springer-Verlag New York Berlin Heidelberg

Ne Set

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

© Springer-Verlag Borlin Hoidelberg 1993 Printed in Germany

Typesetting: Camera ready by author Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr. 45/3140-543210 - Printed on acid-free paper

Foreword

The papers contained in this volume were presented at the fourth annual symposium on Combinatorial Pattern Matching, held June 2-4, 1993 in Padova, Italy. They were selected from 34 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. 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 three meetings were held at the University of Paris in 1990, at the University of London in 1991, and at the University of Arizona, Tucson, in 1992. The first two meetings were informal and no proceedings were produced. The proceedings of the third meeting appeared as Volume 644 in this series.

Several external referees helped with the selection of papers for CPM 93. The Local Organizing Committee consisted of G. Bilardi, L. Colussi, C. Guerra and L. Toniolo. The Padova Ricerche Consortium provided services and, together with other sponsors, funds for the conference. The efforts of all are gratefully acknowledged. Special thanks are due to F. Bombi for eagerly promoting CPM 93 both within the University of Padova and outside of it.

March 1993

Programm Committee

A. Apostolico, chair	G. Gonnet
M. Crochemore	D.S. Hirschberg
A. Ehrenfeucht	U. Manber
A.S. Fraenkel	E.W. Myers
Z. Galil	M.S. Waterman

BIBLIOTHEQUE DU CERIST

۰.

Table of Contents

A Linear Time Pattern Matching Algorithm Between a String and a Tree Tatsuya Akutsu	1
Tight Comparison Bounds for the String Prefix-Matching Problem Dany Breslauer, Livio Colussi and Laura Toniolo	11
3-D Docking of Protein Molecules Daniel Fischer, Raquel Norel, Ruth Nussinov, and Haim J. Wolfson	20
Minimal Separators of Two Words Emmanuelle Garel	35
Covering a String Costas S. Iliopoulos, Dennis W.G. Moore and Kunsoo Park	54
On the Worst-Case Behaviour of Some Approximation Algorithms for the Shortest Common Supersequence of k Strings Robert W. Irving and Campbell B. Fraser	63
An Algorithm for Locating Non-Overlapping Regions of Maximum Alignment Score Sampath K. Kannan and Eugene W. Myers	74
Exact and Approximation Algorithms for the Inversion Distance Between Two Chromosomes John Kececioglu and David Sankoff	87
The Maximum Weight Trace Problem in Multiple Sequence Alignment John Kececioglu	106
An Algorithm for Approximate Tandem Repeats Gad M. Landau and Jeanette P. Schmidt	120
Two Dimensional Pattern Matching in a Digitized Image Gad M. Landau and Uzi Vishkin	134
Analysis of a String Edit Problem in a Probabilistic Framework Guy Louchard and Wojciech Szpankowski	152
Detecting False Matches in String Matching Algorithms S. Muthukrishnan	164

On Suboptimal Alignments of Biological Sequences Dalit Naor and Douglas Brutlag	i 7 9
A Fast Filtration Algorithm for the Substring Matching Problem Pavel A. Pevzner and Michael S. Waterman	197
A Unifying Look at d-Dimensional Periodicities and Space Coverings Mireille Régnier and Ladan Rostami	215
Approximate String-Matching over Suffix Trees Esko Ukkonen	228
Multiple Sequence Comparison and n-Dimensional Image Reconstruction Martin Vingron and Pavel A. Pevzner	243
A New Editing Based Distance Between Unordered Labeled Trees Kaizhong Zhang	254

A Linear Time Pattern Matching Algorithm Between a String and a Tree

Tatsuya AKUTSU *

Mechanical Engineering Laboratory, 1-2 Namiki, Tsukuba, Ibaraki, 305 Japan.

Abstract. In this paper, we describe a linear time algorithm for testing whether or not there is a path of a tree T(|V(T)| = n) that coincides with a string s(|s| = m). In the algorithm, O(n/m) vertices are selected from V(T) such that any path of length more than m - 2 must contain at least one of the selected vertices. A search is performed using the selected vertices as 'bases.' A suffix tree is used effectively in the algorithm. Although the size of the alphabet is assumed to be bounded by a constant in this paper, the algorithm can be applied to the case of unbounded alphabets by increasing the time complexity to $O(n \log n)$.

Keywords: subtree, subgraph isomorphism, string matching, suffix tree, graph algorithms

1 Introduction

The subgraph isomorphism problem is famous and important in computer science. It is the problem of testing whether or not there is a subgraph of T isomorphic to S when the graphs of S and T are given. It is important for practical applications as well. In particular, many heuristic algorithms have been developed for database systems in chemistry [13, 14].

In general, the problem was proved to be NP-complete [6]. However, polynomial time algorithms have been developed in special cases [10, 12]. When the graphs are simple paths, the problem is reduced to the string matching problem, for which several linear time algorithms have been developed [3, 7]. When the graphs are restricted to trees, the problem is solved in $O(n^{2.5})$ time [4]. Moreover, if the vertex degree of two input trees is bounded by a constant, the problem is solved in $O(n^2)$ time. If the graphs are rooted trees such that the labels of children of each node are

^{*} The author thanks to Prof. Tomio Hirata in Nagoya University for helpful comments. This research was partially supported by the Grand-in-Aid for Scientific Research on Priority Areas, "Genome Informatics", of the Ministry of Education, Science and Culture of Japan.

distinct, whether there is an $o(n^2)$ time algorithm or not had been an open problem for a long time. However, it was solved confirmatively by Kosaraju [8] and the result was improved by Dubiner et.al. [5]. However, as far as we know, there is no $o(n^2)$ time algorithm for undirected trees or rooted trees such that children of a node may have identical labels even if the vertex degree is bounded. In this paper, we show a linear time algorithm for a special case of the problem, that is, the case where S is a path and T is an undirected tree. Moreover, T may have a node such that adjacent vertices have identical labels.

In this paper, T denotes an input undirected tree with labeled vertices and $s = s^1 s^2 \cdots s^m$ denotes an input string of length m. We do not assume that labels of vertices adjacent to the same vertex are different. Although vertices are assumed to be labeled, the result can also be applied to the case of labeled edges. For a graph G, V(G) denotes the set of vertices and E(G) denotes the set of edges. n denotes the number of the vertices of the tree T (i.e., n = |V(T)|). For a vertex v, label(v) denotes the label associated with v. We assume that the size of the alphabet is bounded by a constant. The problem is to test whether or not there is a vertex disjoint path (v_1, v_2, \cdots, v_m) in T such that $label(v_1) label(v_2) \cdots label(v_m) = s$. This paper describes an O(n) time algorithm for this problem.

Of course, a rooted tree version of the problem, that is, the case where T is a rooted tree and only the paths which do not connect sibling nodes are allowed, is trivially solved in linear time [5] by using a linear time substring matching algorithm [3, 7] with backtracking. However, this method does not seem to work for the problem of this paper. The linear time algorithm which we developed here is based on a different idea: O(n/m) vertices are selected from V(T) such that any path of length more than m-2 must contain at least one of the selected vertices. From each of the selected vertices, a search is performed with traversing the suffix tree associated with s.

2 Suffix Tree

In this section, we give an overview of the well-known data structure, suffix tree [2, 11]. The suffix tree is used in on-line string matching for a large fixed text. Moreover, it is applied to a variety of pattern matching problems [1, 5, 9].

Let $s = s^1 s^2 \cdots s^m$ be a string, |s| denotes the length of s, s_i denotes the suffix of s which starts from s^i , s^{-1} denotes the reversed string of s and s_i^{-1} denotes $(s^{-1})_i$. For a string s and a character x, sx denotes the concatenation of s and "x". We assume without loss of generality that the special character '#' does not appear in s. The suffix tree SUF_s associated with s is the rooted tree with m leaves and at most m-1 internal nodes such that

- Each edge is associated with a substring of s#.
- Sibling edges must have labels whose first character is distinct.
- Each leaf is associated with a distinct position of s.
- The concatenation of the labels on the path from the root to a leaf l_i describes $(s#)_i$.

It is known that the size of a suffix tree is O(m) and a suffix tree can be constructed in O(m) time if the size of the alphabet is bounded by a constant [11]. It is easy to