Journal of Automata, Languages and Combinatorics 8 (2003) 4, 579–591

COMPUTING ALL REPEATS USING SUFFIX ARRAYS[1,2]

FRANTIŠEK FRANĚK, WILLIAM F. SMYTH[3] and YUDONG TANG
Algorithms Research Group, Department of Computing and Software
McMaster University, Hamilton, Ontario, Canada
e-mail: {franek, smyth}@mcmaster.ca

ABSTRACT

We describe an algorithm that identifies *all* the repeating substrings (tandem, overlapping, and split) in a given string $\boldsymbol{x} = \boldsymbol{x}[1..n]$. Given the suffix arrays of $\boldsymbol{x}$ and of the reversed string $\widehat{\boldsymbol{x}}$, the algorithm requires $\Theta(n)$ time for its execution and represents its output in $\Theta(n)$ space, either as a reduced suffix array (called an NE array) or as a reduced suffix tree (called an NE tree). The output substrings $\boldsymbol{u}$ are *nonextendible* (NE); that is, any extension of some occurrence of $\boldsymbol{u}$ in $\boldsymbol{x}$, either to the left or to the right, yields a string ($\lambda\boldsymbol{u}$ or $\boldsymbol{u}\lambda$) that is unequal to the same extension of some other occurrence of $\boldsymbol{u}$. Thus the number of substrings output is the minimum required to identify all the repeating substrings in $\boldsymbol{x}$. The output can be used in a straightforward way to identify only repeating substrings that satisfy some proximity or minimum length condition.

Keywords: Repeats, suffix arrays, string algorithms, substring

1. Introduction

The computation of all the repeating substrings in a given string $\boldsymbol{x} = \boldsymbol{x}[1..n]$ is a problem with various application areas, most notably data compression, cryptography, and computational biology. For repeating substrings that are *tandem* (that is, *repetitions*), several $O(n \log n)$ algorithms [1, 4, 11] were discovered about 20 years ago; more recently, a repetitions algorithm [9] was published that, at least theoretically, executes in $\Theta(n)$ time in the common case that the alphabet is *indexed* – that is, treatable as a range of integers $1..\alpha \in O(n)$.

These successes with repetitions have encouraged researchers to seek algorithms that efficiently compute all repeating substrings $\boldsymbol{u}$, including in addition to tandem occurrences those that are *split* (of the form $\boldsymbol{uvu}$ for some nonempty $\boldsymbol{v}$) and *overlapping* (such as $\boldsymbol{u} = abaab$ in $\boldsymbol{x} = abaabaab$). The following definitions permit this problem to be stated more precisely.

---

[1]Full version of a lecture presented at the *Thirteenth Australasian Workshop on Combinatorial Algorithms* (Kingfisher Bay Resort, Fraser Island, Queensland, Australia, July 7 – 10, 2002).

[2]Supported in part by grants from the Natural Sciences and Engineering Research Council of Canada.

[3]School of Computing, Curtin University, Perth, Australia.