Journal of Discrete Algorithms 32 (2015) 6–13

Contents lists available at ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

Inferring an indeterminate string from a prefix graph

Ali Alatabbi [a], M. Sohel Rahman [b,*,1], W.F. Smyth [c,d,2]

[a] *Department of Informatics, King's College London, United Kingdom*
[b] *Department of Computer Science & Engineering, Bangladesh University of Engineering & Science, Bangladesh*
[c] *Algorithms Research Group, Department of Computing & Software, McMaster University, Canada*
[d] *School of Engineering & Information Technology, Murdoch University, Australia*

ARTICLE INFO

Article history:
Available online 24 December 2014

Keywords:
String
Indeterminate string
Generalized string
Degenerate string
Prefix table
Prefix graph
Reverse engineering

ABSTRACT

An ***indeterminate string*** (or, more simply, just a ***string***) $\boldsymbol{x} = \boldsymbol{x}[1..n]$ on an alphabet Σ is a sequence of nonempty subsets of Σ. We say that $\boldsymbol{x}[i_1]$ and $\boldsymbol{x}[i_2]$ ***match*** (written $\boldsymbol{x}[i_1] \approx \boldsymbol{x}[i_2]$) if and only if $\boldsymbol{x}[i_1] \cap \boldsymbol{x}[i_2] \neq \emptyset$. A ***feasible array*** is an array $\boldsymbol{y} = \boldsymbol{y}[1..n]$ of integers such that $\boldsymbol{y}[1] = n$ and for every $i \in 2..n$, $\boldsymbol{y}[i] \in 0..n-i+1$. A ***prefix table*** of a string $\boldsymbol{x}$ is an array $\boldsymbol{\pi} = \boldsymbol{\pi}[1..n]$ of integers such that, for every $i \in 1..n$, $\boldsymbol{\pi}[i] = j$ if and only if $\boldsymbol{x}[i..i+j-1]$ is the longest substring at position i of $\boldsymbol{x}$ that matches a prefix of $\boldsymbol{x}$. It is known from [6] that every feasible array is a prefix table of some indeterminate string. A ***prefix graph*** $\mathcal{P} = \mathcal{P}_{\boldsymbol{y}}$ is a labelled simple graph whose structure is determined by a feasible array $\boldsymbol{y}$. In this paper we show, given a feasible array $\boldsymbol{y}$, how to use $\mathcal{P}_{\boldsymbol{y}}$ to construct a lexicographically least indeterminate string on a minimum alphabet whose prefix table $\boldsymbol{\pi} = \boldsymbol{y}$.

1. Introduction

In the extensive literature of stringology/combinatorics on words, a "string" or "word" has usually been defined as a sequence of individual elements of a distinguished set Σ called an "alphabet". Nevertheless, going back as far as the groundbreaking paper of Fischer and Paterson [9], more general sequences, defined instead on *subsets* of Σ, have also been considered. The more constrained model introduced in [9] restricts entries in a string to be either elements of Σ (subsets of size 1) or Σ itself (subsets of size $\sigma = |\Sigma|$); these have been studied in recent years as "strings with don't cares" [14], also "strings with holes" or "partial words" [3]. The unconstrained model, which allows arbitrary nonempty subsets of Σ, has also attracted significant attention, often because of applications in bioinformatics: such strings have variously been called "generalized" [1], "indeterminate" [13], or "degenerate" [15].

In this paper we study strings in their full generality, hence the following definitions:

Definition 1. Suppose a set Σ of symbols (called the ***alphabet***) is given. A ***string*** $\boldsymbol{x}$ on Σ of ***length*** $n = |\boldsymbol{x}|$ is a sequence of $n \geq 0$ nonempty finite subsets of Σ, called ***letters***; we represent $\boldsymbol{x}$ as an array $\boldsymbol{x}[1..n]$. If $n = 0$, $\boldsymbol{x}$ is called the ***empty string*** and denoted by $\boldsymbol{\varepsilon}$; if for every $i \in 1..n$, $\boldsymbol{x}[i]$ is a subset of Σ of size 1, $\boldsymbol{x}$ is said to be a ***regular string***.

* Corresponding author.
E-mail addresses: ali.alatabbi@kcl.ac.uk (A. Alatabbi), msrahman@cse.buet.ac.bd (M. Sohel Rahman), smyth@mcmaster.ca (W.F. Smyth).
[1] Partially supported by a Commonwealth Fellowship and an ACU Titular Fellowship. Currently on a Sabbatical Leave from BUET.
[2] Supported in part by Grant No. 5-36797 of the Natural Sciences & Engineering Research Council of Canada.

http://dx.doi.org/10.1016/j.jda.2014.12.006
1570-8667/© 2014 Elsevier B.V. All rights reserved.

Definition 2. Suppose we are given two strings $\boldsymbol{x}$ and $\boldsymbol{y}$ and integers $i \in 1..|\boldsymbol{x}|$, $j \in 1..|\boldsymbol{y}|$. We say that $\boldsymbol{x}[i]$ and $\boldsymbol{y}[j]$ ***match*** (written $\boldsymbol{x}[i] \approx \boldsymbol{y}[j]$) if and only if $\boldsymbol{x}[i] \cap \boldsymbol{y}[j] \neq \emptyset$. Then $\boldsymbol{x}$ and $\boldsymbol{y}$ ***match*** ($\boldsymbol{x} \approx \boldsymbol{y}$) if and only if $|\boldsymbol{x}| = |\boldsymbol{y}|$ and $\boldsymbol{x}[i] \approx \boldsymbol{y}[i]$ for every $i \in 1..|\boldsymbol{x}|$.

Note that matching is not necessarily transitive: $a \approx \{a, b\} \approx b$, but $a \not\approx b$.

Definition 3. The ***prefix table*** (also ***prefix array***)[3] of a string $\boldsymbol{x} = \boldsymbol{x}[1..n]$ is the integer array $\boldsymbol{\pi_x} = \boldsymbol{\pi_x}[1..n]$ such that for every $i \in 1..n$, $\boldsymbol{\pi_x}[i]$ is the length of the longest prefix of $\boldsymbol{x}[i..n]$ that matches a prefix of $\boldsymbol{x}$. Thus for every prefix table $\boldsymbol{\pi_x}$, $\boldsymbol{\pi_x}[1] = n$. When there is no ambiguity, we write $\boldsymbol{\pi} = \boldsymbol{\pi_x}$.

The prefix table is an important data structure for strings: it identifies all the borders, hence all the periods, of every prefix of $\boldsymbol{x}$ [6]. It was originally introduced to facilitate the computation of repetitions in regular strings [16], see also [20]; and for regular strings, prefix table and border array are equivalent, since each can be computed from the other in linear time [5]. For general strings, the prefix table can be computed in compressed form in $O(n^2)$ time using $\Theta(n/\sigma)$ bytes of storage space [21], where $\sigma = |\Sigma|$. Two examples follow, adapted from [6]:

$$\begin{array}{rcccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \boldsymbol{x_1} = & a & c & a & g & a & c & a & t \\ \boldsymbol{\pi_1} = & 8 & 0 & 1 & 0 & 3 & 0 & 1 & 0 \end{array} \tag{1}$$

$$\begin{array}{rcccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \boldsymbol{x_2} = & \{a,c\} & \{g,t\} & \{a,g\} & \{a,c,g\} & g & c & \{a,t\} & a \\ \boldsymbol{\pi_2} = & 8 & 0 & 4 & 2 & 0 & 3 & 1 & 1 \end{array} \tag{2}$$

Since clearly every position $i \in 2..n$ in a prefix table $\boldsymbol{\pi}$ must satisfy $0 \le \boldsymbol{\pi}[i] \le n - i + 1$, the following definition is a natural one:

An array $\boldsymbol{y} = \boldsymbol{y}[1..n]$ of integers is said to be a ***feasible array*** if and only if $\boldsymbol{y}[1] = n$ and for every $i \in 2..n$, $\boldsymbol{y}[i] \in 0..n - i + 1$.

An immediate consequence of Definition 3 is the following:

Lemma 1. *(See [6].) Let $\boldsymbol{x} = \boldsymbol{x}[1..n]$ be a string. An integer array $\boldsymbol{y} = \boldsymbol{y}[1..n]$ is the prefix table of $\boldsymbol{x}$ if and only if for each position $i \in 1..n$, the following two conditions hold:*

(a) $\boldsymbol{x}[1..\boldsymbol{y}[i]] \approx \boldsymbol{x}[i..i + \boldsymbol{y}[i] - 1]$;
(b) if $i + \boldsymbol{y}[i] \le n$, then $\boldsymbol{x}[y[i] + 1] \not\approx \boldsymbol{x}[i + \boldsymbol{y}[i]]$.

Then the following fundamental result establishes the important connection between strings and feasible arrays:

Lemma 2. *(See [6].) Every feasible array is the prefix table of some string.*

In view of this lemma, we say that a feasible array is ***regular*** if it is the prefix array of a regular string. We are now able to state the goal of this paper as follows: for a given feasible array $\boldsymbol{y} = \boldsymbol{y}[1..n]$, not necessarily regular, construct a string $\boldsymbol{x}$ on a minimum alphabet whose prefix table $\boldsymbol{\pi_x} = \boldsymbol{y}$ — the "reverse engineering" problem for the prefix table in its full generality. In fact, we do somewhat more: we construct a lexicographically least such string, in a sense to be defined in Section 2.

The first reverse engineering problem in stringology was stated and solved in [11,10], where an algorithm was described to compute a lexicographically least regular string whose border array was a given integer array — or to return the result that no such regular string exists. Many other similar constructions have since been published, related to other stringological data structures but always specific to regular strings (see [2,8,12,18], among others). [19] was the first paper to consider the more general problem of inferring an indeterminate string from a given data structure (specifically, border array, suffix array and LCP array). Although solving such problems does not yield immediate applications, nevertheless solutions provide a deeper understanding of the combinatorial many–many relationship between strings and the various data structures developed from them (see for example [17], where canonical strings corresponding to given border arrays are identified and efficiently generated for use as test data).

For prefix tables and regular strings, the reverse engineering problem was solved in [7], where a linear-time algorithm was described to return a lexicographically least regular string $\boldsymbol{x}$ whose prefix table is the given feasible array $\boldsymbol{y}$, or an

[3] We prefer "table" because of the possible confusion with "suffix array", a completely different data structure.