Discrete Applied Mathematics 215 (2016) 41–46

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

V-Order: New combinatorial properties & a simple comparison algorithm[☆]

Ali Alatabbi[a,*], Jacqueline W. Daykin[a,b], Juha Kärkkäinen[c], M. Sohel Rahman[d], W.F. Smyth[a,e,f]

[a] Department of Informatics, King's College London, UK
[b] Department of Computer Science, Royal Holloway, University of London, UK
[c] Department of Computer Science, University of Helsinki, Finland
[d] AℓEDA Group, Department of Computer Science & Engineering, Bangladesh University of Engineering & Science, Bangladesh
[e] Algorithms Research Group, Department of Computing & Software, McMaster University, Canada
[f] School of Engineering & Information Technology, Murdoch University, Western Australia, Australia

ARTICLE INFO

Article history:
Received 28 January 2016
Received in revised form 21 June 2016
Accepted 4 July 2016
Available online 5 August 2016

Keywords:
Combinatorics
Experiments
Lexorder
Linear
On-line algorithm
Optimal
String comparison
V-comparison
V-order

ABSTRACT

V-order is a global order on strings related to Unique Maximal Factorization Families (UMFFs), themselves generalizations of Lyndon words. V-order has recently been proposed as an alternative to lexicographic order in the computation of suffix arrays and in the suffix-sorting induced by the Burrows–Wheeler transform. Efficient V-ordering of strings thus becomes a matter of considerable interest. In this paper we discover several new combinatorial properties of V-order, then explore the computational consequences; in particular, a fast, simple on-line V-order comparison algorithm that requires no auxiliary data structures.

1. Introduction

Our interest in V-order [6] arises out of a generalization of Lyndon words called Unique Maximal Factorization Families (UMFFs) [7] whose combinatorial properties were explored in [8]. More recently, several papers [9,10,1,2] have investigated the combinatorial properties of V-order itself, with particular emphasis on algorithms for V-order string comparison. This latter topic becomes interesting because of recent work [11] showing that V-order can be used as an alternative to lexicographic order in the computation of suffix arrays [13] and in the suffix-sorting induced by the Burrows–Wheeler transform [14].

In this paper we first prove a collection of combinatorial properties of V-order making clear that V-order comparison of strings can be done in a manner analogous to lexicographic comparison, by simply traversing the string from left to right.

[☆] This work was supported in part by Grant No. 8180 of the Natural Sciences & Engineering Research Council of Canada.

[*] Corresponding author.
E-mail addresses: ali.alatabbi@kcl.ac.uk (A. Alatabbi), jackie.daykin@kcl.ac.uk, jackie.daykin@rhul.ac.uk (J.W. Daykin), juha.karkkainen@cs.helsinki.fi (J. Kärkkäinen), msrahman@cse.buet.ac.bd (M. Sohel Rahman), smyth@mcmaster.ca (W.F. Smyth).

http://dx.doi.org/10.1016/j.dam.2016.07.006
0166-218X/© 2016 Elsevier B.V. All rights reserved.

We then go on to propose a very simple on-line V-order comparison algorithm that requires no additional data structures and that moreover is much faster than any of its predecessors.

In Section 2 we introduce basic concepts, including V-order and V-form, and state two lemmas important for our development. Section 3 provides new combinatorial results that constitute the theoretical background for our new V-order comparison algorithm. The algorithm itself is specified in Section 4, and the results of computer experiments are discussed in Section 5. Most of the combinatorial results in this paper appeared first in [3].

2. Preliminaries

Consider a finite ordered *alphabet* Σ; that is, a set of *letters* of cardinality $\sigma = |\Sigma|$. A *string* is a sequence of zero or more letters over Σ. A string $\boldsymbol{x}$ of *length* $|\boldsymbol{x}| = n$ is represented as an array $\boldsymbol{x}[1..n]$, where $\boldsymbol{x}[i] \in \Sigma$ for $1 \le i \le n$. The set of all nonempty strings over Σ is denoted by Σ^+. The *empty string* of length zero is denoted by $\boldsymbol{\varepsilon}$, with $\Sigma^* = \Sigma^+ \cup \boldsymbol{\varepsilon}$, often written $\boldsymbol{x}[i..j]$ with $j < i$. A string $\boldsymbol{w}$ is a *substring*, or *factor*, of $\boldsymbol{x}$ if $\boldsymbol{x} = \boldsymbol{uwv}$ with $\boldsymbol{u}, \boldsymbol{v} \in \Sigma^*$; then $\boldsymbol{u}$ is a *prefix* and $\boldsymbol{v}$ a *suffix* of $\boldsymbol{x}$. A subsequence of $\boldsymbol{y}$ is a string $\boldsymbol{x}$ defined by $\boldsymbol{x}_k = \boldsymbol{y}_{n_k}$, where $n_1 < n_2 < \cdots$ is an increasing sequence of indices. A string $\boldsymbol{x}$ is a proper subsequence of $\boldsymbol{y}$ if $\boldsymbol{x}$ is a subsequence of $\boldsymbol{y}$ and $\boldsymbol{x} \neq \boldsymbol{y}$. A string $\boldsymbol{y}$ is a *rotation* of $\boldsymbol{x}[1..n]$, written $\boldsymbol{y} = R_i(\boldsymbol{x})$, if $\boldsymbol{y} = \boldsymbol{x}[i..n]\boldsymbol{x}[1..i-1]$ for some $1 \le i < n$ (for $i = 1, \boldsymbol{y} = \boldsymbol{x}$). Given two strings $\boldsymbol{x}$ and $\boldsymbol{y}$ with $|\boldsymbol{x}| < |\boldsymbol{y}|$, $\boldsymbol{x}$ is *lexicographically less* than $\boldsymbol{y}$ ($\boldsymbol{x} < \boldsymbol{y}$) if and only if

- $\boldsymbol{x}$ is a prefix of $\boldsymbol{y}$; or
- $\boldsymbol{x}$ and $\boldsymbol{y}$ have a common prefix $\boldsymbol{u}$ of length $\ell = |\boldsymbol{u}|$ and $\boldsymbol{x}[\ell+1] < \boldsymbol{y}[\ell+1]$.

Then $\boldsymbol{x}[1..n]$ is a *Lyndon word* if and only if $\boldsymbol{x} < R_i(\boldsymbol{x})$ for every $1 < i < n$. For further stringological definitions, theory and algorithmics see [5].

Theorem 1 (*[4]*). *Any word $\boldsymbol{w}$ can be written uniquely as a non-increasing sequence $\boldsymbol{w} = \boldsymbol{u}_1\boldsymbol{u}_2 \ldots \boldsymbol{u}_k$ of Lyndon words.*

This famous theorem was followed a quarter-century later by an equally famous algorithm [12] that computed the decomposition $\boldsymbol{u}_1 \ge \boldsymbol{u}_2 \ge \cdots \ge \boldsymbol{u}_k$ in time $\mathcal{O}(|\boldsymbol{w}|)$. With this background, we now define a non-lexicographic global order, V-order, and explore its properties.

Let $\boldsymbol{x} = x_1x_2 \ldots x_n$ be a string over Σ. Define $h \in \{1, \ldots, n\}$ by $h = 1$ if $x_1 \le x_2 \le \cdots \le x_n$; otherwise, by the unique value such that $x_{h-1} > x_h \le x_{h+1} \le x_{h+2} \le \cdots \le x_n$. Let $\boldsymbol{x}^* = x_1x_2 \ldots x_{h-1}x_{h+1} \ldots x_n$, where the star * indicates deletion of the letter x_h. Write $\boldsymbol{x}^{s*}$ for $(\ldots(\boldsymbol{x}^*)^* \ldots)^*$ with $s \ge 0$ stars. Let $g = \max\{x_1, x_2, \ldots, x_n\}$, and let k be the number of occurrences of g in $\boldsymbol{x}$. Then the sequence $\boldsymbol{x}, \boldsymbol{x}^*, \boldsymbol{x}^{2*}, \ldots$ ends with $g^k, \ldots, g^2, g^1, g^0 = \boldsymbol{\varepsilon}$. In the *star tree* each string $\boldsymbol{x}$ over Σ labels a vertex, and there is a directed edge from $\boldsymbol{x}$ to $\boldsymbol{x}^*$, with $\boldsymbol{\varepsilon}$ as root.

Definition 1. We define V-*order* $\prec$ between distinct strings $\boldsymbol{x}$, $\boldsymbol{y}$. First $\boldsymbol{x} \prec \boldsymbol{y}$ if $\boldsymbol{x}$ is in the path $\boldsymbol{y}, \boldsymbol{y}^*, \boldsymbol{y}^{2*}, \ldots, \boldsymbol{\varepsilon}$. If $\boldsymbol{x}, \boldsymbol{y}$ are not in a path, there exist smallest s, t such that $\boldsymbol{x}^{(s+1)*} = \boldsymbol{y}^{(t+1)*}$. Put $\boldsymbol{s} = \boldsymbol{x}^{s*}$ and $\boldsymbol{t} = \boldsymbol{y}^{t*}$; then $\boldsymbol{s} \neq \boldsymbol{t}$ but $|\boldsymbol{s}| = |\boldsymbol{t}| = m$ say. Let $j \in 1..m$ be the greatest integer such that $\boldsymbol{s}[j] \neq \boldsymbol{t}[j]$. If $\boldsymbol{s}[j] < \boldsymbol{t}[j]$ in Σ then $\boldsymbol{x} \prec \boldsymbol{y}$.

Example 1. Using the natural ordering of integers, if $\boldsymbol{x} = 32\,415$, then $\boldsymbol{x}^* = 3245$, $\boldsymbol{x}^{2*} = 345$, $\boldsymbol{x}^{3*} = 45$ and so $45 \prec 32\,415$.

Definition 2 (*[6,7,9,10]*). The V-**form** of a string $\boldsymbol{x}$ is defined as

$$V_k(\boldsymbol{x}) = \boldsymbol{x} = \boldsymbol{x_0}g\boldsymbol{x_1}g \ldots \boldsymbol{x_{k-1}}g\boldsymbol{x_k}$$

for strings $\boldsymbol{x_i}$, $i = 0, 1, \ldots, k$, where g is the largest letter in $\boldsymbol{x}$—thus we suppose that g occurs exactly k times. For clarity, when more than one string is involved, we use the notation $g = \mathcal{L}_{\boldsymbol{x}}$, $k = \mathcal{C}_{\boldsymbol{x}}$.

Lemma 1 (*[6,7,9,10]*). *Suppose we are given distinct strings $\boldsymbol{x}$ and $\boldsymbol{y}$ with corresponding V-forms as follows:*

$$\boldsymbol{x} = \boldsymbol{x}_0\mathcal{L}_{\boldsymbol{x}}\boldsymbol{x}_1\mathcal{L}_{\boldsymbol{x}}\boldsymbol{x}_2 \cdots \boldsymbol{x}_{j-1}\mathcal{L}_{\boldsymbol{x}}\boldsymbol{x}_j, \quad (1)$$

$$\boldsymbol{y} = \boldsymbol{y}_0\mathcal{L}_{\boldsymbol{y}}\boldsymbol{y}_1\mathcal{L}_{\boldsymbol{y}}\boldsymbol{y}_2 \cdots \boldsymbol{y}_{k-1}\mathcal{L}_{\boldsymbol{y}}\boldsymbol{y}_k, \quad (2)$$

where $j = \mathcal{C}_{\boldsymbol{x}}$, $k = \mathcal{C}_{\boldsymbol{y}}$. Then $\boldsymbol{x} \prec \boldsymbol{y}$ if, and only if, one of the following conditions holds:

(C1) $\mathcal{L}_{\boldsymbol{x}} < \mathcal{L}_{\boldsymbol{y}}$
(C2) $\mathcal{L}_{\boldsymbol{x}} = \mathcal{L}_{\boldsymbol{y}}$ *and* $\mathcal{C}_{\boldsymbol{x}} < \mathcal{C}_{\boldsymbol{y}}$
(C3) $\mathcal{L}_{\boldsymbol{x}} = \mathcal{L}_{\boldsymbol{y}}$, $\mathcal{C}_{\boldsymbol{x}} = \mathcal{C}_{\boldsymbol{y}}$ *and* $\boldsymbol{x}_h \prec \boldsymbol{y}_h$, *where* $h \in \{0 \ldots \max(j,k)\}$ *is the least integer such that* $\boldsymbol{x}_h \neq \boldsymbol{y}_h$.

Lemma 2 (*[9,10]*). *For given strings $\boldsymbol{v}$ and $\boldsymbol{x}$, if $\boldsymbol{v}$ is a proper subsequence of $\boldsymbol{x}$, then $\boldsymbol{v} \prec \boldsymbol{x}$.*

Example 2. We compare the two orderings for a set of English words over the ordered Roman alphabet:

Lexorder ($<$): *catastrophe* $<$ *sop* $<$ *strop* $<$ *strophe* $<$ *top*.
The words are scanned from left to right, seeking the first difference.
V-order ($\prec$): *sop* $\prec$ *top* $\prec$ *strop* $\prec$ *strophe* $\prec$ *catastrophe*.

The first V-order comparison is determined by Lemma 1(C1), the following three by the useful Lemma 2.