Available online at www.sciencedirect.com

SCIENCE DIRECT®

Information Processing Letters 93 (2005) 269–274

Information Processing Letters

www.elsevier.com/locate/ipl

Substring search and repeat search using factor oracles

Ryoichi Kato, Osamu Watanabe *

Tokyo Institute of Technology, Department of Mathematical and Computing Science, Meguro-ku, Ookayama 2-12-1 W8-25, Tokyo 152-8552, Japan

Received 5 April 2004; received in revised form 22 November 2004

Available online 29 December 2004

Communicated by K. Iwama

Abstract

We present some simple but useful properties of factor oracles, and propose fast algorithms for indexed full-text search and finding repeated substrings. Some experiments are given to demonstrate the performance of our algorithms.

Keywords: Algorithms; Automaton; Suffix-trees; Substring search; Repetition search

1. Introduction

Factor oracles [1] are deterministic acyclic automata built from a given text. Factor oracles are more space economical and easy to implement than similar machineries such as suffix-trees or suffix-automata. Unfortunately, however, unlike a suffix-tree or a suffix-automaton, which precisely accepts only substrings, of an input text, a factor oracle may accept strings not in the text. Thus, in some situations, factor oracles cannot be used directly as an index for full-text search. Here we show some simple properties, which enable us to use a factor oracle as an index of a given text, avoiding to accept patters not in the text. Also for the application of using factor oracles to find repeats (see, e.g., [4]), we derive some property and propose an improvement of existing algorithms [3,5].

Before starting our discussion, we give here necessary notations, notions, and properties on factor oracles, most of which have been introduced/proved in [2]. We ask the reader to refer it for the detail. We will use standard notions and notations on strings such as $|u|$, the length of a string u, etc. For any strings u and v, we write $u \preceq v$ if u is a *suffix* of v, and write $u \precneqq v$ if $u \preceq v$ and $u \neq v$.

A factor oracle is constructed from a given *text* string over a finite alphabet Σ. Throughout this note, we use p and m to denote a given text and its length $|p|$. A *substring* (or, a *factor*) of p is a string w such that $p = uwv$ for some $u, v \in \Sigma^*$; in particular, for any i and j, $1 \leqslant i \leqslant j \leqslant m$, we use $p[i..j]$ to denote the *substring* of p appearing from the ith character to the jth character. We say that w *appears at position* i

* Corresponding author.
E-mail address: watanabe@is.titech.ac.jp (O. Watanabe).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2004.11.011

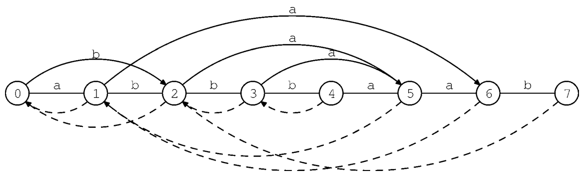

Fig. 1. Oracle("abbbaab"). Internal transitions are edges on a straight line (omitting arrows). Solid arcs left to right on the upper side are external transitions. Dashed arcs right to left on the down side are suffix links.

if w is a suffix of $p[1..i]$, and i is called an *end position* of the occurrence of w. For any string $w \in \Sigma^*$ and integer i, $1 \leqslant i \leqslant m$, define

$$\mathsf{endpos}(w) = \{i \mid w = p[i - |w| + 1..i]\},$$

and

$$\mathsf{repet}(i) = \text{the longest suffix of } p[1..i] \text{ that occurs more than once in } p[1..i].$$

That is, $\mathsf{endpos}(w)$ is the set of end positions of the occurrences of w, and $\mathsf{repet}(i)$ is the longest repeated suffix of $p[1..i]$. *Substring search* or *full-text search* is to compute the smallest/all elements of $\mathsf{endpos}(w)$ for a given w. *Repeat search* is to compute $\mathsf{repet}(i)$ and its end position before i, for all i, $1 \leqslant i \leqslant n$. We will propose algorithms for these tasks.

A *factor oracle* $\mathsf{Oracle}(p)$ (for a given text p) is a deterministic acyclic automaton consisting of $m + 1$ states, m internal transitions, and at most $m - 1$ external transitions; see Fig. 1 for an example. An *internal transition* from state i to state $i + 1$ is a transition that occurs at state i when reading $p[i]$, the ith character of p. On the other hand, for any state i and any $\sigma \in \Sigma$, an *external transition* is defined as follows: Define $\mathsf{min}(i)$ to be the shortest string that reaches to i from state 0 by internal and (so far defined) external transitions. Then an external transition from i to j via σ is defined if j is the smallest index such that $\mathsf{min}(i)\sigma$ appears at j, i.e., $\mathsf{min}(i)\sigma$ is a suffix of $p[1..j]$; the transition is undefined if no such j exists. In the factor oracle $\mathsf{Oracle}(p)$, state 0 is the initial state and all states are accepting states. Thus, all strings reaching to some state from state 0 are accepted by $\mathsf{Oracle}(p)$. Note that $\mathsf{min}(i)$ is the shortest string that is accepted at state i.

Below we state some of the basic properties of factor oracles from [2] that will be used in our discussion.

Lemma 1. *For any substring w of p, w is accepted by $\mathsf{Oracle}(p)$. Indeed, it is accepted at some i such that $i \leqslant j$, for all $j \in \mathsf{endpos}(w)$.*

Lemma 2. *For any i, $\mathsf{min}(i)$ appears at i; indeed, i is the smallest element of $\mathsf{endpos}(\mathsf{min}(i))$.*

Lemma 3. *For any i, any string accepted at i has $\mathsf{min}(i)$ as a suffix. On the other hand, if $\mathsf{Oracle}(p)$ accepts some string with $\mathsf{min}(i)$ as a suffix, then the string must be accepted at j such that $j \geqslant i$.*

2. Exact recognition using factor oracles

Lemma 1 guarantees that any string rejected by factor oracle $\mathsf{Oracle}(p)$ is not a substring of p. On the other hand, there is a case that $\mathsf{Oracle}(p)$ accepts a string not appearing in p. Here we show some simple properties, with which we can check efficiently whether an accepted string indeed appears as a substring. We will make use of "suffix link" that has been introduced [1] to construct factor oracles efficiently. Note that it has been shown, see, e.g., [2], that $\mathsf{Oracle}(p)$ including its suffix links can be constructed from p in time $\mathrm{O}(|p|)$.

For any state $i > 0$ of $\mathsf{Oracle}(p)$, its *suffix link* $\mathsf{S}(i)$ is i', where i' is the state that string $\mathsf{repet}(i)$ is accepted at; see Fig. 1 for an example. We leave $\mathsf{S}(0)$ undefined. By definition, for any $i > 0$, $\mathsf{S}(i)$ is uniquely determined; it is easy to see (formally from Lemma 1) that $\mathsf{S}(i) < i$ for any state $i > 0$. Then from any state $i > 0$, by following the suffix links, we eventually reach to the state 0; that is,

$$i > \mathsf{S}(i) > \mathsf{S}\big(\mathsf{S}(i)\big) > \cdots > \mathsf{S}\big(\cdots \mathsf{S}(i) \cdots\big) = 0.$$

We call this sequence of states a *suffix path*, and define $\mathsf{SP}(i)$ to be the set of states on the suffix path from i.

The following property, though natural and in fact easy to obtain, will be useful for our analysis.

Lemma 4. *For any i,* $\mathsf{repet}(\mathsf{S}(i)) \precneqq \mathsf{min}(\mathsf{S}(i)) \preceq \mathsf{repet}(i) \precneqq \mathsf{min}(i)$.

Proof. Note that both $\mathsf{repet}(i)$ and $\mathsf{min}(i)$ appear at i; $\mathsf{min}(i)$ appears for the first time (from the left) at i (by Lemma 2), whereas $\mathsf{repet}(i)$ must have appeared before i (by definition). Hence, it cannot be the case that