Journal of Discrete Algorithms 14 (2012) 91–106

Contents lists available at SciVerse ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

Worst-case efficient single and multiple string matching on packed texts in the word-RAM model

Djamal Belazzougui [1]

LIAFA, Univ. Paris Diderot, Paris 7, 75205 Paris Cedex 13, France

ARTICLE INFO

Article history:
Available online 7 December 2011

Keywords:
Independent set
Fixed-parameter tractability
Hereditary class of graphs
Modular decomposition

ABSTRACT

In this paper, we explore worst-case solutions for the problems of single and multiple matching on strings in the word-RAM model with word length w. In the first problem, we have to build a data structure based on a pattern p of length m over an alphabet of size σ such that we can answer to the following query: given a text T of length n, where each character is encoded using $\log\sigma$ bits return the positions of all the occurrences of p in T (in the following we refer by *occ* to the number of reported occurrences). For the multi-pattern matching problem we have a set S of d patterns of total length m and a query on a text T consists in finding all positions of all occurrences in T of the patterns in S. As each character of the text is encoded using $\log\sigma$ bits and we can read w bits in constant time in the RAM model, we assume that we can read up to $\Theta(w/\log\sigma)$ consecutive characters of the text in one time step. This implies that the fastest possible query time for both problems is $O(n\frac{\log\sigma}{w} + occ)$. In this paper we present several different results for both problems which come close to that best possible query time. We first present two different linear space data structures for the first and second problem: the first one answers to single pattern matching queries in time $O(n(\frac{1}{m} + \frac{\log\sigma}{w}) + occ)$ while the second one answers to multiple pattern matching queries to $O(n(\frac{\log d + \log y + \log\log m}{y} + \frac{\log\sigma}{w}) + occ)$ where y is the length of the shortest pattern. We then show how a simple application of the four Russian technique permits to get data structures with query times independent of the length of the shortest pattern (the length of the only pattern in case of single string matching) at the expense of using more space.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

The problems of string pattern matching and multiple string pattern matching are classical algorithmic problems in the area of pattern matching. In the multiple string matching problem, we have to preprocess a dictionary of d strings of total length m characters over an alphabet of size σ so that we can answer to the following query: given any text of length n, find all occurrences in the text of any of the d strings. In the case of single string matching, we simply have $d = 1$.

The textbook solutions for the two problems are the Knuth–Morris–Pratt [22] (KMP for short) automaton for the single string matching problem and the Aho–Corasick [1] automaton (AC for short) for the multiple string matching problem. The AC automaton is actually a generalization of the KMP automaton. Both algorithms achieve $O(n + occ)$ query time (where *occ* denotes the number of reported occurrences) using $O(m\log m)$[2] bits of space[3] (both automatons are encoded using $O(m)$ pointers occupying $\log m$ bits each). The query time of both algorithms is in fact optimal if the matching is restricted to read

[1] This work is supported by the French ANR-2010-COSI-004 project MAPPI.

[2] In this paper $\log x$ is defined as $\lceil\log_2(x+2)\rceil$.

[3] In this paper we quantify the space usage in bits rather than in words as is usual in other papers.

1570-8667/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.jda.2011.12.011

all the characters of the text one by one. However as it was noticed in previous works, in many cases it is actually possible to avoid reading all the characters of the text and hence achieve a better performance. This stems from the fact that by reading some characters at certain positions in the text, one could conclude whether a match is possible or not without the need to read all the characters. This has led to various algorithms with so-called sublinear query time assuming that the characters of the patterns and/or the text are drawn from some random distribution. The first algorithm which exploited that fact was the Boyer–Moore algorithm [7]. Subsequently other algorithms with provably average-optimal performance were devised. Most notably the BDM and BNDM for single string matching and the multi-BDM [13,11] and multi-BNDM [26] for multiple string matching. Those algorithms achieve $O(n\frac{\log m}{m\log\sigma} + occ)$ time for single string matching (which is optimal according to the lower bound shown in [32]) and $O(n\frac{\log d+\log y}{y\log\sigma} + occ)$ time for multiple string matching, where y is the length of the shortest string in the set. Still in the worst case those algorithms may have to read all the text characters and thus have $\Omega(n + occ)$ query time (actually many of those algorithms have an even worse query time in the worst-case, namely $\Omega(nm + occ)$).

A general trend has appeared in the last two decades when many papers have appeared trying to exploit the power of the word-RAM model to speed-up and/or reduce the space requirement of classical algorithms and data structures. In this model, the computer operates on words of length w and usual arithmetic and logic operations on the words all take one unit of time.

In this paper we focus on the worst-case bounds in the RAM model with word length w. That is we try to improve on the KMP and AC in the RAM model assuming that we have to read all the characters of the text which are assumed to be stored in a contiguous area in memory using $\log\sigma$ bits per characters. That means that it is possible to read $\Theta(w/\log\sigma)$ consecutive characters of the text in $O(1)$ time. Thus given a text of length n characters, an optimal algorithm should spend $O(n\frac{\log\sigma}{w} + occ)$ time to report all the occurrences of matching patterns in the text. The main result of this paper is a worst-case efficient algorithm whose performance is essentially the addition of a term similar to the average optimal time presented above plus the time necessary to read all the characters of the text in the RAM model. Unlike many other papers, we only assume that $w = \Omega(\log(n+m))$, and not necessarily that $w = \Theta(\log(n+m))$. That is we only assume that a pointer to the manipulated data (the text and the patterns), fit in a memory word but the word length w can be arbitrarily larger than $\log m$ or $\log n$. This assumption makes it possible to state time bounds which are independent of m and n, implying larger speedups for small values of m and n.

In his paper Fredriksson presents a general approach [18] which can be applied to speed-up many pattern matching algorithms. This approach which is based on the notion of super-alphabet relies on the use of tabulation (four Russian technique). If this approach is applied to our problems of single and multiple string matching queries, given an available precomputed space t, we can get a $\log_\sigma(t/m)$ factor speedup. In his paper [6], Bille presented a more space efficient method for single string matching queries which accelerates the KMP algorithm to answer to queries in time $O(\frac{n}{\log_\sigma n} + occ)$ using $O(n^\varepsilon + m\log m)$ bits of space for any constant ε such that $0 < \varepsilon < 1$. More generally, the algorithm can be tuned to use an additional amount t of tabulation space in order to provide a $\log_\sigma t$ factor speedup.

At the end of his paper, Bille asked two questions: the first one was whether it is possible to get an acceleration proportional to the machine word length w (instead of $\log n$ or $\log t$) using linear space only. The second one was whether it is possible to obtain similar results for the multiple string matching problem. We give partial answers to both questions. Namely, we prove the following two results:

1. Our first result states that for d strings of minimal length y, we can construct an index which occupies linear space and answers to queries in time $O(n(\frac{\log d+\log y+\log\log m}{y} + \frac{\log\sigma}{w}) + occ)$. This result implies that we can get a speedup factor $\frac{w}{(\log d+\log w)\log\sigma}$ if $y \geqslant \frac{w}{\log\sigma}$ and get the optimal speedup factor $\frac{w}{\log\sigma}$ if $y \geqslant (\log d + \log w)\frac{w}{\log\sigma}$.
2. Our second result implies that for d patterns of arbitrary lengths and an additional t bits of memory, we can obtain a factor $\frac{\log_\sigma t}{\log d+\log\log_\sigma t+\log\log m}$ speedup using $O(m\log m + t)$ bits of memory.

Our first result compares favorably to Bille's and Fredriksson approaches as it does not use any additional tabulation space. In order to obtain any significant speedup, the algorithms of Bille and Fredriksson require a substantial amount of space t which is not guaranteed to be available. Even if such an amount of space was available, the algorithm could run much slower in case $m \ll t$ as modern hardware is made of memory hierarchies, where random access to large tables which do not fit in the fast levels of the hierarchy might be much slower than access to small data which fit in faster levels of the hierarchy.

Our second result is useful in case the shortest string is very short and thus, the first result do not provide any speedup. The result is slightly less efficient than that of Bille [6] for single string matching, being a factor $\log\log_\sigma t + \log\log m$ slower (compared to the $\log_\sigma t$ speedup of Bille's algorithm). However, our second result efficiently extends to multiple string matching queries, while Bille's algorithms seems not to be easily extensible to multiple string matching queries.

The third and fourth results in this paper are concerned with single string matching, where we can have solutions with a better query time than what can be obtained by using the first and second result for matching a single pattern. In particular our results imply the following:

Download English Version:

https://daneshyari.com/en/article/431645

Download Persian Version:

https://daneshyari.com/article/431645

Daneshyari.com