Improved Approximate String Matching
Using Compressed Suffix Data Structures

Tak-Wah Lam · Wing-Kin Sung · Swee-Seong Wong

Received: 9 February 2006 / Accepted: 21 August 2006 / Published online: 1 November 2007
© Springer Science+Business Media, LLC 2007

Abstract

Approximate string matching is about finding a given string pattern in a
text by allowing some degree of errors. In this paper we present a space efficient data
structure to solve the 1-mismatch and 1-difference problems. Given a text $T$ of length
$n$ over an alphabet $A$, we can preprocess $T$ and give an $O(n\sqrt{\log n \log |A|})$-bit space
data structure so that, for any query pattern $P$ of length $m$, we can find all 1-mismatch
(or 1-difference) occurrences of $P$ in $O(|A|m \log \log n + occ)$ time, where $occ$ is the
number of occurrences. This is the fastest known query time given that the space of
the data structure is $o(n \log^2 n)$ bits.

The space of our data structure can be further reduced to $O(n \log |A|)$ with the
query time increasing by a factor of $\log^\epsilon n$, for $0 < \epsilon \leq 1$. Furthermore, our solution
can be generalized to solve the $k$-mismatch (and the $k$-difference) problem in
$O(|A|^k m^k (k + \log \log n) + occ)$ and $O(\log^\epsilon n (|A|^k m^k (k + \log \log n) + occ)$ time
using an $O(n \sqrt{\log n \log |A|})$-bit and an $O(n \log |A|)$-bit indexing data structures, re-
spectively. We assume that the alphabet size $|A|$ is bounded by $O(2^{\sqrt{\log n}})$ for the
$O(n \sqrt{\log n \log |A|})$-bit space data structure.

1 Introduction

Consider a text $T$ of length $n$ and a pattern $P$ of length $m$, both strings over an
alphabet $A$. The approximate string matching problem is to find all approximate oc-

T.-W. Lam
Department of Computer Science, The University of Hong Kong, Hong Kong, Hong Kong
e-mail: twlam@cs.hku.hk

W.-K. Sung · S.-S. Wong (✉)
School of Computing, National University of Singapore, Singapore, Singapore
e-mail: wongss@comp.nus.edu.sg

W.-K. Sung
e-mail: ksung@comp.nus.edu.sg
currences of $P$ in $T$. Depending on the definition of “error,” this problem has two variations: (1) The $k$-difference problem is to find all occurrences of $P$ in $T$ that have edit distance at most $k$ from $P$ (edit distance is the minimum number of character insertions, deletions and replacements to convert one string to another); and (2) The $k$-mismatch problem is to find all occurrences of $P$ in $T$ that have Hamming distance at most $k$ from $P$ (Hamming distance is the minimum number of character replacements to convert one string to another). Both $k$-difference and $k$-mismatch problems are well-studied and they found applications in many areas including computational biology, text retrieval, multi-media data retrieval, pattern recognition, signal processing, handwriting recognition, etc.

In the past, most of the works focus on the online version of the problem, that is, both the text and the pattern are not known in advance. This version of the problem can be solved by dynamic programming in $O(nm)$ time. Landau and Vishkin [11] gave a solution whose running time depends on $k$, the number of allowed “errors.” They solved the problem in $O(nk)$ time and $O(m)$ space. Amir et al. [2] improved upon the result for $k$-mismatch, to give an $O(n\sqrt{k\log k})$ time solution. We refer to [14] for a comparison study on various existing techniques.

Recently, people are interested in the offline approximate matching problem, in which we can preprocess the text $T$ and build some indexing data structure so that any query can be answered in a shorter time. Jokinen and Ukkonen [10] were the first to treat the approximate offline matching problem. Since then, many different approaches have been proposed. (Please refer to [17] for a brief survey.) Some techniques are fast on the average [3, 15, 16, 18, 23, 24]. However, they incur a query time complexity depending on $n$, i.e., in the worst case, they are inefficient even if the pattern is very short and $k$ is as small as one. The first solution with query time complexity independent of $n$ is proposed by Ukkonen [26]. When $k = 1$ (that is, 1-mismatch or 1-difference problem), Cobbs [5] gave the result of using $O(n\log n)$ bits and having $O(|A|m^2 + occ)$ query time. Later, Amir et al. [1] proposed an $O(n\log^3 n)$-bit indexing data structure with $O(m\log n\log\log n + occ)$ query time. Then, Buchsbaum et al. [4] proposed another indexing data structure which uses $O(n\log^2 n)$ bits so that every query can be solved in $O(m\log\log n + occ)$ time. Cole et al. [6] further improved the query time. They gave an $O(n\log^2 n)$-bit data structure so that both the 1-mismatch and the 1-difference problems can be solved in $O(m + n\log\log n + occ)$ time, respectively. Recently, motivated by the indexing of long genomic sequences, Trinh et al. [25] improves upon the space-efficiency. They proposed two data structures of size $O(n\log n)$ bits and $O(n)$ bits with query time $O(|A|m\log n + occ)$ and $O(|A|m\log^2 n + occ\log n)$, respectively.

Some of the above results can be generalized for $k > 1$. Cobbs’ $O(n\log n)$-bit indexing data structure can answer both $k$-mismatch and $k$-difference queries in $O(m^{k+1}|A|^k + occ)$ time [5]. Cole et al. [6] proposed an $O(n\frac{\log n}{k!}\log\log n + m + occ)$-bit indexing data structure with query times of $O\left(\frac{c_1\log n}{k!}\log\log n + m + occ\right)$ and $O\left(\frac{c_2\log n}{k!}\log\log n + m + 3^k\cdot occ\right)$ for the $k$-mismatch and $k$-difference problems, respectively, where $c_1, c_2, c_3$ are constants with $c_2 > c_1$. Trinh et al. [25] gave $O(n\log n)$-bit and $O(n\log |A|)$-