Department of Computer Science
Series of Publications C
Report C-2004-1

Sub-quadratic Algorithm for Weighted k-Mismatches Problem

Veli Mäkinen

University of Helsinki
Finland

Sub-quadratic Algorithm for Weighted k-Mismatches Problem

Veli Mäkinen

Department of Computer Science
P.O. Box 26, FIN-00014 University of Helsinki, Finland
vmakinen@cs.Helsinki.FI

Technical report, Series of Publications C, Report C-2004-1
Helsinki, February 2004, 2 pages

Abstract

The classical *approximate string matching problem* is to find approximate occurrences of a *pattern* string inside a *text* string. In *k-mismatches problem*, the approximate occurrences are such text substrings that differ at most in k positions from the pattern. It is well-known that the k-mismatches problem can be solved in $o(mn)$ time, where m is the pattern length and n is the text length. Algorithms with complexity $O(kn)$ also exist. A generalization of the problem is the *weighted k-mismatches problem*: Each character pair is assigned a weight, and an approximate occurrence is such that the sum of the weights between matched pattern/text characters is at most k. This problem can be solved in $O(mn)$ using the trivial algorithm; no asymptotically better algorithm exists. Muthukhrishnan (CPM 1995) asks whether there is a sub-quadratic algorithm for this problem. We answer this question affirmatively in the case that the alphabet is of fixed size σ; we give an algorithm with running time $O(mn/\log_\sigma n)$.

Computing Reviews (1998) Categories and Subject Descriptors:
F.2.2 Analysis of Algorithms and Problem Complexity: Nonnumerical Algorithms and Problems—pattern matching, sorting and searching, geometrical problems and computations

General Terms:
Algorithms, Theory

Additional Key Words and Phrases:
Combinatorial pattern matching, Four Russians technique

1 Introduction

Let $P = p_1 p_2 \cdots p_m$ be a *pattern* string and $S = s_1 s_2 \cdots s_n$ a *text* string over an alphabet Σ. Let $f : \Sigma \times \Sigma \to \mathbb{R}$ be a weighting function. The *weighted k-mismatches problem* is to find all positions j of S such that

$$\sum_{i=1}^{m} f(p_i, s_{j+i-1}) \leq k, \tag{1}$$

where k is a given threshold parameter.

The trivial algorithm for this problem is to test each text position separately. This takes $O(mn)$ time. For the case that the weights are integers, Bertossi et al. [2] give an $O(kn)$ time algorithm generalizing an earlier algorithm of Landau and Vishkin [4] for the unit cost case. Their algorithm also allows insertions and deletions of characters with constant weight. However, the trivial algorithm is still asymptotically faster, since k is not bounded by m in the weighted case.

Muthukhrishnan [5] states as an open problem whether there is an $o(mn)$ time algorithm for the weighted k-mismatches problem. This paper provides such an algorithm for the case that the alphabet is of fixed size σ; the time complexity of the algorithm is $O(mn/\log_\sigma n)$.

2 The Algorithm

The algorithm uses two well-known techniques: Incremental index computation [3] and the "Four Russians" technique [1].

2.1 Incremental Index Computation

We may assume w.l.o.g. that our alphabet is $\Sigma = \{0, 1, \ldots, \sigma - 1\}$. (If this is not the case, we can make a bijective alphabet remapping in time $O((m+n)\log m)$.) Let $v_\ell(S_j)$ give the *rank* of the string $S_j = s_j s_{j+1} \cdots s_{j+\ell-1}$ in the lexicographic order of all strings of length ℓ (over the alphabet Σ). We have

$$v_\ell(S_j) = s_j \times \sigma^{\ell-1} + s_{j+1} \times \sigma^{\ell-2} + \cdots + s_{j+\ell-1}. \tag{2}$$

It follows that

$$v_\ell(S_{j+1}) = (v_\ell(S_j) - s_j \times \sigma^{\ell-1})\sigma + s_{j+\ell}, \tag{3}$$

thus the rank of the next substring can be computed in constant time.

2.2 Four Russians Technique

Consider a string X of length $\ell = \log_\sigma n$. Let $g(X)$ be an additive function that assigns a real value for X; additive in a sense that $g(X) = g(\alpha) + g(\beta)$ if $X = \alpha\beta$. Our task is to store values $g(X)$ for all possible strings of length ℓ so that each value $g(X)$ can be retrieved in constant time. We can only afford to use $o(n)$ time for preprocessing. The trick is to store answers for strings of length $\ell/2$ (assume w.l.o.g. that ℓ is even). There are $\sigma^{\ell/2} = \sigma^{(\log_\sigma n)/2} = \sqrt{n}$ such strings. Then $g(X)$ can be computed in constant time: $g(X) = g(x_1 x_2 \cdots x_{\ell/2}) + g(x_{\ell/2+1} x_{\ell/2+2} \cdots x_\ell)$.

2.3 Combining the Techniques

We store a table $F[0\ldots\sqrt{n}-1][0\ldots 2m/\log_\sigma n-1]$ such that $F[q][r]$ gives the partial sum of weighted differences between a text substring and a pattern substring. More formally, let $X = x_1 x_2 \cdots x_{\ell/2}$ be the $(q+1)$th string in the lexicographic order of all strings of length $\ell/2$, where $\ell = \log_\sigma n$. That is, $v_{\ell/2}(X) = q$. Then

$$F[q][r] = \sum_{i=r\times\ell/2+1}^{(r+1)\times\ell/2} f(p_i, x_{i-r\times\ell/2}). \tag{4}$$

The size of $F[][]$ is $O(\sqrt{n}m/\ell)$ and we need $O(\ell)$ time to compute each cell. Overall it takes $O(\sqrt{n}m)$ time to fill the table.

Searching for the occurrences of the pattern can be done as follows. We maintain $2m/\ell$ variables $w_0, w_1, \ldots, w_{2m/\ell-1}$ such that at position j of the text holds

$$w_r = v_{\ell/2}(S_{j+r\times\ell/2}). \tag{5}$$

Then Equation (1) can be rewritten as

$$\sum_{r=0}^{2m/\ell-1} F[w_r][r] \le k. \tag{6}$$

Thus the testing at position j can be done in $O(m/\ell)$ time. To proceed to the next position $j+1$, we need to update each variable w_r to the value $v_{\ell/2}(S_{j+1+r\times\ell/2})$. Updating a variable can be done in constant time using Equation (3).

Overall the algorithm takes $O(\sqrt{n}m + nm/\log_\sigma n)$ time.

References

[1] V. L. Arlazarov, E. A. Dinic, M. A. Kronrod, and I. A. Faradzev. On economic construction of the transitive closure of a directed graph. *Dokl. Acad. Nauk. SSSR 194*, 487–488, 1970 (in Russian). English translation in *Soviet Math. Dokl. 11*, 1209–1210, 1975.

[2] A. Bertossi, F. Luccio, E. Lodi, and L. Pagli. String Matching with Weighted Errors. *Theoretical Computer Science*, 73:319–328, 1990.

[3] R. Karp ja M. Rabin. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31:249–260, 1987.

[4] G. Landau and U. Vishkin. Introducing efficient parallelism into approximate string matching and a new serial algorithm. In *Proc. 18th Ann. ACM Symp. on Theory of Computing*, Berkeley, CA, pp. 220–230, 1986.

[5] S. Muthukrishnan. New Results and Open Problems Related to Non-Standard Stringology. In *Proc. Combinatorial Pattern Matching (CPM 1995)*, pages 298–317, 1995.