Mathematical Problems of Computer Science 54, 116--121, 2020.

UDC 519.688

Fuzzy String Matching Using a Prefix Table

Armen H. Kostanyan

IT Educational and Research Center
Yerevan State University, Armenia
e-mail: armko@ysu.am

Abstract

The string matching problem (that is, the problem of finding all occurrences of a pattern in the text) is one of the well-known problems in symbolic computations with applications in many areas of artificial intelligence. The most famous algorithms for solving it are the finite state machine method and the Knuth-Morris-Pratt algorithm (KMP).

In this paper, we consider the problem of finding all occurrences of a fuzzy pattern in the text. Such a pattern is defined as a sequence of fuzzy properties of text characters. To construct a solution to this problem, we introduce a two-dimensional prefix table, which is a generalization of the one-dimensional prefix array used in the KMP algorithm.

Keywords: KMP algorithm, Approximate string matching, Fuzzy string matching

1. Introduction

String matching is a well-known computational problem, the earliest references to which date to the 1960s. Interest in this problem really grew with the publication of the Boyer-Moor (BM) [1] and Knuth-Morris-Pratt (KMP) [2] algorithms in the 1970s. Since then, quite a lot of exact string matching algorithms have been proposed using one or another modification of BM or KMP.
Along with investigations on exact string matching, there have also been investigations on approximate string matching that looked at the problem of finding a generic pattern in the text. Unlike the regular pattern, specified as a list of characters, the generic pattern is rather some description of the substring to be found. Research on approximate string matching has focused on distance-based string matching [3], string matching using meta-characters in patterns, and more generally, string matching using a regular expression as a pattern [4]. A detailed survey of these works is given in the monograph [5].

In this paper, we formulate a fuzzy string matching problem, in which the pattern is represented as a sequence of fuzzy characters, and the problem of finding all occurrences of such a pattern in a text with a given accuracy is defined. This problem was investigated in [6], where a non-deterministic transition system was constructed to describe the possibilities of processing the given text in order to find all occurrences of a fuzzy pattern in it, and an efficient algorithm for determining some occurrences was proposed. In contrast, here we propose an efficient algorithm to determine all occurrences of a fuzzy pattern in the text, mimicking the KMP algorithm.

The paper is organized as follows.

Section 2 presents the problem definitions. Section 3 introduces the concept of a two-dimensional prefix table, which generalizes the notion of a one dimensional prefix array used in the KMP algorithm. The text processing algorithm using the prefix table is presented in Section 4. The complexity of the proposed algorithm is analyzed in Section 5. Finally, the conclusion summarizes the obtained results.

2. The Fuzzy String Matching Problem

Suppose that $(L, \vee, \wedge, 0, 1)$ is a finite lattice with the smallest element 0 and the largest element 1. According to [7], the *fuzzy subset* A of the universal set U is defined by the membership function μ_A: $U \to L$ that associates with each element x from U the number $\mu_A(x)$ in L, representing the *grade of membership* of x in A. A fuzzy subset A from U can be represented by the additive form

$$A = \sum_{x \in U} x / \mu_A(x).$$

We say that an element x definitely belongs to A if $\mu_A(x) = 1$, and it definitely does not belong to A if $\mu_A(x) = 0$. On the contrary, if $0 < \mu_A(x) < 1$, we say that x belongs to A with degree $\mu_A(x)$.

Let us define a *fuzzy symbol* α over the alphabet Σ to be a fuzzy subset of Σ. Given a character $x \in \Sigma$, we say that x matches α with the grade $\mu_\alpha(x)$. We define the fuzzy pattern $P[1 .. m]$ as a sequence of fuzzy symbols of length m.

For a given text $T[1 .. n]$, a fuzzy pattern $P[1 .. m]$ and a threshold λ, we formulate the *fuzzy string matching problem* as the problem of finding all positions w (or, *valid shifts*), $1 \le w \le n\text{-}m\text{+}1$, in the text such so that

$$\mu_{P[k]}\ (T[w + k - 1]) \ \ge\ \lambda \text{ for all } k,\ 1 \le k \le m.$$

3. Prefix Table

Let $x = x[1 .. q]$ be an array of text symbols of length q and $\lambda \ge 0$. Let us define the x-border of $P = P[1..m]$ as a subarray $P[1 .. k]$ ($k < q$) such that

$$\mu_{P[i]}\ (x[q - k + i]) \ge \lambda \text{ for all } i,\ 1 \le i \le k,$$

(that is, the last k characters of x match the first k symbols of P with degree of at least λ).

Let $\mathrm{LB}_P(x)$ denote the longest x-border of P.

***Example* 3.1:** Choose $\Sigma = \{\mathbf{1, 2, 3, 4, 5}\}$, $L = \{ 0, 0.25, 0.5, 0.75, 1 \}$ and define the fuzzy symbols **S** (*small*), **M** (*middle*) and **L** (*large*) as follows:

$\mathbf{S} = \mathbf{1}/1 + \mathbf{2}/0.75 + \mathbf{3}/0.5 + \mathbf{4}/0.25 + \mathbf{5}/0,$
$\mathbf{M} = \mathbf{1}/0 + \mathbf{2}/0.75 + \mathbf{3}/1 + \mathbf{4}/0.75 + \mathbf{5}/0,$
$\mathbf{L} = \mathbf{1}/0 + \mathbf{2}/0.25 + \mathbf{3}/0.5 + \mathbf{4}/0.75 + \mathbf{5}/1.$

Then, for x = **14252**, P = **SLMLS** and $\lambda = 0.75$, the following statement holds

$$\mathrm{LB}_P(x) = \mathbf{SLM}.$$

Let us define the *prefix table* as a matrix $\pi[1 \,..\, m, 1 \,..\, n]$ such that

$$\pi[q, i] = \begin{cases} 0, & \text{if } q = 1 \\ |\mathrm{LB}_P(T[i \,..\, i+q\text{-}1])|, & \text{if } 2 \le q \le m,\ 1 \le i \le n-q+1 \\ \textit{undefined}, & \text{if } i+q\text{-}1 > n \end{cases}$$

Claim 1: For all $2 \le q \le m$, $1 \le i \le n-q+1$

$$\pi[q, i] = \begin{cases} q - 1, & \text{if } \pi[q\text{-}1, i] = q\text{-}2 \text{ and } \mu_{P[q\text{-}1]}(T[i+q\text{-}1]) \ge \lambda \\ \\ \pi[q\text{-}1, i+1], & \text{otherwise} \end{cases} \qquad (1)$$

Proof: Follows from the following statements:

- $[|\mathrm{LB}_P(T[i \,..\, i+q\text{-}1])| = q\text{-}1] \Rightarrow [\mu_{P[s]}(T[i+s]) \ge \lambda$ for all s, $1 \le s \le q\text{-}2] \Rightarrow$
  $\Rightarrow [|\mathrm{LB}_P(T[i \,..\, q\text{-}1])| = q\text{-}2, \mu_{P[q\text{-}1]}(T[i+q\text{-}2]) \ge \lambda] \Rightarrow$
  $\Rightarrow [\pi[q\text{-}1, i] = q\text{-}2, \mu_{P[q\text{-}1]}(T[i+q\text{-}2]) \ge \lambda] \Rightarrow [\pi[q, i] = q\text{-}1]$.
- $[|\mathrm{LB}_P(T[i \,..\, i+q\text{-}1])| < q\text{-}1] \Rightarrow [|\mathrm{LB}_P(T[i \,..\, i+q\text{-}1])| = |\mathrm{LB}_P(T[i+1 \,..\, i+q\text{-}1])|]$
- $\Rightarrow [\pi[q, i] = \pi[q\text{-}1, i+1]]$.

The formula (1) allows us to construct the prefix table by filling it from top to bottom and from left to right. The procedure below provides additional details.

Compute-Prefix-Table(P, T, λ)

```
1  $m = P.length$, $n = T.length$
2  let $\pi[1..m, 1..n]$ be the prefix table to be filled accordingly
3  for $i = 1$ to $n$
4      $\pi[1, i] = 0$
5  for $q = 2$ to $m$                //substring length
6      for $i = 1$ to $n-q+1$        //position in the text
7          if $\pi[q-1, i] == q - 2$ and $\mu_{P[q-1]}(T[i+q-1]) \ge \lambda$
8              $\pi[q, i] = q - 1$
9          else $\pi[q, i] = \pi[q-1, i+1]$
10 return $\pi$
```

***Example* 3.2:** Suppose that

P = **MSMSLM,** T = **223141325422414251**,

where **S**, **M** and **L** are the fuzzy symbols defined in *Example 3.1*. The matrix in Fig. 1 presents a prefix table built by the **Compute-Prefix-Table** algorithm based on P, T and $\lambda = 0.75$.

$$\begin{array}{c|cccccccccccccccccc|}
 & \mathbf{2} & \mathbf{2} & \mathbf{3} & \mathbf{1} & \mathbf{4} & \mathbf{1} & \mathbf{3} & \mathbf{2} & \mathbf{5} & \mathbf{4} & \mathbf{2} & \mathbf{2} & \mathbf{4} & \mathbf{1} & \mathbf{4} & \mathbf{2} & \mathbf{5} & \mathbf{1} \\
\mathbf{M} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{S} & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & - \\
\mathbf{M} & 1 & 2 & 1 & 2 & 1 & 2 & 0 & 1 & 2 & 2 & 1 & 2 & 1 & 2 & 0 & 0 & - & - \\
\mathbf{S} & 2 & 3 & 2 & 3 & 2 & 0 & 1 & 2 & 3 & 3 & 2 & 3 & 2 & 0 & 0 & - & - & - \\
\mathbf{L} & 3 & 4 & 3 & 4 & 0 & 1 & 2 & 3 & 3 & 4 & 3 & 4 & 0 & 0 & - & - & - & - \\
\mathbf{M} & 4 & 3 & 4 & 5 & 1 & 2 & 3 & 3 & 4 & 5 & 4 & 5 & 0 & - & - & - & - & -
\end{array}$$

Fig. 1. Prefix table for P = **MSMSLM,** T = **223141325422414251**.

4. Text Processing using a Prefix Table

The following procedure is an adaptation of the KMP algorithm to solve the fuzzy strings matching problem using a prefix table.

KMP-FUZZY-STRING-MATCHING (*P*, *T*, *λ*)

```
1   $m = P.length$, $n = T.length$,
2   $\pi$ = Compute-Prefix-Table($P$, $T$, $\lambda$)
3   $q = 0$                        // number of characters matched
4   for $i = 1$ to $n$
5       while $q > 0$ and $\mu_{P[q+1]}(T[i]) < \lambda$
6           $q = \pi[q, i - q]$
7       if $\mu_{P[q+1]}(T[i]) \geq \lambda$
8           $q = q + 1$
9       if $q == m$                //the entire pattern matches
10          Print( $i - q + 1$ )
11          $q = \pi[q, i - q + 1]$
```

***Example* 4.1.** Fig. 2 illustrates the fuzzy string matching process for T, P and λ from *Example 3.2* and the prefix table in Fig. 1.

```
    1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18
T:  2  2  3  1  4  1  3  2  5  4  2  2  4  1  4  2  5  1
    1  2  3  4  5  #
P:  M  S  M  S  L  M
          1  2  3  4  #
          M  S  M  S  L
                1  2  3  4  5  6
                M  S  M  S  L  M  - matching
                               1  2  3  #
                               M  S  M  S
                                  1  2  3  4  5  6
                                  M  S  M  S  L  M  - matching
                                        1  2  3  4  5  #
                                        M  S  M  S  L  M
                                                       #
                                                       M
```

Fig. 2. Fuzzy string matching process.

5. Analysis

To estimate the complexity of the **KMP-FUZZY-STRING-MATCHING** algorithm, let us take advantage of the potential method. Define the potential before every execution of the body of the **for** loop equal to q, which is initially 0 and never becomes negative. Note that the **while** loop has an amortized cost of O(1) since the total number of executions of the assignment statement in line 6 does not exceed the potential value before this loop is executed. The two subsequent **if** statements obviously have O(1) amortized complexities. Thus, the amortized complexity of the execution of the body of the **for** loop is O(1), which gives O(n) complexity for the *processing phase*. The complexity of the *preprocessing* phase represented by the **Compute-Prefix-Table** procedure is obviously O(mn).

The algorithm uses O(mn) extra memory needed to represent the prefix table π.

5. Conclusion

The problem of finding occurrences of a fuzzy pattern in a given text with a given accuracy has been considered in this paper. A generalization of the KMP string matching algorithm is proposed to solve this problem. Like the KMP algorithm, the proposed algorithm consists of preprocessing and processing phases. The preprocessing phase creates a two-dimensional prefix table that is used in text processing.

For a pattern of length m and a text of length n, the preprocessing phase takes O(mn) time, and the processing phase takes O(n) time. The extra space used by the algorithm is O(mn).

References

[1] R. S. Boyer and J. S. Moore, "A fast string matching algorithm," *Association for Computing Machinery*, vol. 20, no. 10, pp. 762-772, 1977.

[2] D. Knuth and M. Pratt, "Fast pattern matching in strings," *SIAM J. Comput*. vol. 6, no. 2, pp. 323-350, 1977.

[3] G. Landau and U. Vishkin, "Efficient string matching with k mismatches," TCS, vol. 43, pp. 239-249, 1986.

[4] R. Baeza-Yates and N. Gonzaio, "A faster algorithm for approximate string matching," in *Proc. of Seventh Annual Symp. Combinatorial Pattern Matching*, Spinger-Verlag, 1996, pp. 1-23.

[5] B. Smyth, "Computing Patterns in Strings", Addison-Wesley UK, 2003.

[6] A. Kostanyan, "Fuzzy String Matching with Finite Automata", in *Proceedings on 2017 IEEE Conference CSIT-2017*, Yerevan, Armenia, pp. 25-29. IEEE Press, USA 2018.

[7] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning-I," *Information Sciences,* vol. 8, pp. 199-249, 1975.

Submitted 16.07.20, accepted 12.11.20.

Տողում ենթատողի ոչ հստակ որոնում նախածանցների աղյուսակի օգտագործմամբ

Արմեն Հ. Կոստանյան
ՏՏ կրթական և հետազոտական կենտրոն
Երևանի պետական համալսարան, Հայաստան
e-mail: armko@ysu.am

Ամփոփում

Տողում ենթատողի որոնման խնդիրը արհեստական բանականության տարբեր բնագավառներում լայն կիրառում ունեցող խնդիրներից է: Նշված խնդրի լուծման հանրահայտ ալգորիթմներն են՝ վերջավոր ավտոմատների մեթոդը և Կնուտի-Մորիսի-Պրատի (ԿՄՊ) ալգորիթմը:

Տվյալ հոդվածում դիտարկվում է հստակ տողում (տեքստում) ոչ հստակ ենթատողի (շաբլոնի) որոնման խնդիրը: Ոչ հստակ ենթատողը սահմանվում է որպես տեքստի սիմվոլների ոչ հստակ հատկությունների հաջորդականություն: Խնդրի լուծումը կառուցվում է ԿՄՊ ալգորիթմում օգտագործվող նախածանցների միաչափ աղյուսակի երկչափ ընդլայնման եղանակով:

Բանալի բառեր՝ ԿՄՊ ալգորիթմ, տողում ենթատողի մոտավոր որոնում, տողում ենթատողի ոչ հստակ որոնում:

Нечеткий поиск подстроки в строки с использованием таблицы префиксов

Армен Г. Костанян

Образовательный и научный центр ИТ
Ереванский государственный университет, Армения
e-mail: armko@ysu.am

Аннотация

Задача поиска подстроки в строке одна из часто встречающихся задач в области символических вычислений, имеющая многочисленные применения в задачах искусственного интеллекта. К числу наиболее известных алгоритмов для ее решения относятся метод конечных автоматов и алгоритм Кнута-Морриса-Прата (алгоритм КМП).

В настоящей статье рассматривается задача поиска всех вхождений нечеткой подстроки в строке. При этом нечеткая подстрока определяется как последовательность нечетких свойств символов строки (текста). Поставленная задача решается с помощью двумерной таблицы префиксов, являющейся обобщением одномерной таблицы префиксов, используемой в алгоритме КМП.

Ключевые слова: алгоритм КМП, приближенный поиск подстроки в строке, нечеткий поиск подстроки в строке.