

doi: 10.12052/gdutxb.160157

一种基于二进制编码处理的数字保序匹配算法

罗国辉, 林 穗, 姜文超

(广东工业大学 计算机学院, 广东 广州 510006)

摘要: 给出一个数字文本 T 跟一个数字文本段落 P , 在 T 中找出所有跟 P 相同关联顺序的子串称之为数字保序匹配问题. 数字保序匹配问题是检测相似度和预测趋势领域的研究重点, 在股票预测、基金走势、天气预报等方面有广泛应用. 针对保序匹配算法在数字匹配中的效率问题, 提出了基于二进制编码的数字字符匹配方法. 首先, 在保序匹配前先对数字字符进行二进制编码预处理; 然后使用 4 种主流算法实现了保序匹配, 并对匹配时间和复杂度进行对比分析. 实验结果表明: 该处理方法有效提高了数字字符匹配效率, 同时保持了较低的时间复杂度.

关键词: 保序匹配; 二进制编码; 时间复杂度; 字符匹配算法

中图分类号: TP301 **文献标志码:** A **文章编号:** 1007-7162(2017)05-0056-04

A Digital Order-Preserving-Matching Algorithm Based on Binary Coding Processing

Luo Guo-hui, Lin Sui, Jiang Wen-chao

(School of Computers Science, Guangdong University of Technology, Guangzhou 510006, China)

Abstract: Giving a digital text T and a number of text paragraphs P , and then finding in T all the same order with P -related sub-string is called the digital order preserving and matching. Digital order-matching is the focus of research on similarity and prediction trend, widely used in stock forecast, fund trend and weather forecast. Aiming at the efficiency of preserving and matching algorithm in digital matching, a character matching method based on binary coding has been proposed. Firstly, the binary characters are preprocessed before the order matching. Then the four main algorithms are used to achieve the order-preserving and matching, and the matching time and complexity are compared and analyzed. The experimental results show that the proposed method effectively improves the efficiency of digital character matching while maintaining a low time complexity.

Key words: order-preserving; binary-coding; time complexity; character matching algorithm

数字保序匹配问题是检测相似度和预测趋势领域的研究重点. 目前, 字符串匹配技术应用已逐步扩展到股票预测分析, 基金走势, 气象预报等众多领域^[1]. 给出一个长度为 n 的文本 T 及长度为 m 的段落 P ; T , P 都属于有限集合 Z , 所谓字符串匹配是从 P 中找到所有的变化趋势与 T 相同的子字符串. 随着对字符串匹配问题的不断研究, 特别是数字字符匹配在日常生活中变得越来越重要, 如每日股票的成交量及每股的股值动态趋势变化, 日常的天气温度、湿度等的趋势变化都是相应的数值变化^[2-6]. 假设数字字符 $P=(10, 22, 15, 30, 20, 18, 27)$, $T=(22, 85, 79, 24, 42,$

$27, 62, 40, 32, 47, 69, 55, 25)$, 那么就可以得出 T 中与 P 有关联顺序的子串 $u=(24, 42, 27, 62, 40, 32, 47)$.

KMP 方法首先是由 Kubica 等^[4]提出, 该算法最先是在计算顺序边缘表上使用, Kubica 等人通过研究发现从文本中找出所有与段落中的关联关系表所耗时是线性的. 因此, 表明总时间复杂度是线性的. Kim 等^[3]基于前缀表示方法改进了 KMP. 而前缀表示方法又是基于查找前缀中的每个数的秩的方法. 这种方法的时间复杂度为 $O(n\log m)$. Cho 等^[5]基于坏字符规则提出 BMH 方法在 q-grams 问题中使用, 这种方法在文本中查找时因为每当遇到一个坏字符就会跳

过大段的字符,所以算法的平均复杂度是亚线性的.在最坏的情况下时间复杂度为 $O(mn)$. Cho等人提出了一种基于时间复杂度为线性的改进版本^[6],这种版本结合了KMP方法.

对于保序匹配的问题,有学者提出了离线处理方法^[2]和在线处理方法^[3-5, 7-8]. Kubica与Kim等^[3-4]提出了一种基于Knuth-Morris-Pratt (KMP)算法的处理方法. Cho等^[5-6]给出了一种基于坏字符探索的亚线性算法(Boyer-Moore算法^[9]), Belazzougui等^[7]提出了一种改进的亚线性算法来处理保序匹配问题. 本文提出一种新的基于二进制转换与编码的保序匹配算法. 该算法有效提高了数字字符匹配效率和保序匹配质量,且具有线性时间复杂度.

1 相关工作

为了方便后面的论述,这里给出顺序同构及其相关的定义和证明.

定义1 两个字符串 $u = u_1, u_2, \dots, u_m, v = v_1, v_2, \dots, v_m$ 在字符集中有相同的长度,称之为顺序同构^[3-4],如果

$$u_i \leq u_j \Leftrightarrow v_i \leq v_j (1 \leq i, j \leq m),$$

则可表示为 $u \approx v$.

定义2 对字符串 x 的前缀表达式 u_x :

$$u_x[i] = \{j : x[j] \leq x[i], 0 \leq j \leq i\}.$$

引理1 对于两个字符串 x 和 y ,

若 $x \approx y$, 则 $u_x \approx u_y$.

证明 假设 $x \approx y$, $x[i] \leq x[j] \Leftrightarrow y[i] \leq y[j]$, $0 \leq i < j < |x|$. 因此, $u_x = u_y$.

引理2 假如 $x[0 \cdot t] \approx y[0 \cdot t]$. 且 $0 \leq i, j \leq t$, 若 $x[i] < x[j]$, 则 $y[i] < y[j]$, 若 $x[i] = x[j]$, 则 $y[i] = y[j]$.

证明 首先证明第一种情形:(1) 当 $x[i] < x[j]$; 假如 $y[i] \geq y[j]$, 则由定义1可知 $x[i] \geq x[j]$, 这与假设 $x[i] < x[j]$ 矛盾.(2) 当 $x[i] = x[j]$; 由定义1知 $x[i] \leq x[j]$, $y[i] \leq y[j]$, 同样可由顺序同构知, 若 $y[i] \leq y[j]$ 且 $y[j] \leq y[i]$, $y[i] = y[j]$, 则 $x[j] \leq x[i]$, $y[j] \leq y[i]$.

在本文所研究的问题中,研究的目的是从 $T = t_1 t_2 \dots t_n$ 中找出所有的顺序同构字符串 $P = p_1 p_2 \dots p_m$. 首先把数字字符转换成二进制字符,即 $P = p_1 p_2 \dots p_m$ 转换成 $P^* = b_1 b_2 \dots b_{m-1}$. 转换的规则如式(1):

$$t_i = \begin{cases} 1, & t_{i-1} < t_i; \\ 0, & \text{其他.} \end{cases} \quad p_i = \begin{cases} 1, & p_{i-1} < p_i; \\ 0, & \text{其他.} \end{cases} \quad (1)$$

若 $p_i < p_{i+1}$, 则 b_i 为1, 否则 b_i 为0. 在对文本 T, P 进行处理时,会有以下几种特殊情况. 这里采用二进制

向量 E 表示; 如果 $p_{r[i]} < p_{r[i+1]}$, $i < j$, 则 $E[i] = 1$; 若 $p_{r[i]} = p_{r[i+1]}$ 或者 $p_{r[i]} > p_{r[i+1]}$, 则 $E[i] = 0$; 因此在以下情况下本文是不做研究的.

$$t_{j-1+r[i]} > t_{j-1+r[i+1]} \quad (2)$$

$$t_{j-1+r[i]} = t_{j-1+r[i+1]}, E[i] = 0 \quad (3)$$

$$t_{j-1+r[i]} < t_{j-1+r[i+1]}, E[i] = 1 \quad (4)$$

这里 t_j 表示在 T 中的字符.

2 算法介绍与复杂度分析

2.1 KMP算法

本文中介绍的是基于前缀表达式改进的KMP算法. 算法匹配的过程中如果遇到 T, P 不匹配时则进行两个部分操作.(1) 计算模式串移动 $\text{next}[j]$ 后末字符在文本中的位置;(2) 比较文本和模式串在该位置的字符是否匹配, 算法描述:

1. input into n
2. JUDGE;
3. while $i < n$
4. if $T[i] = P[i]$
5. if $j = m - 1$
6. find and count++;
7. $i++$, $j = 0$
8. goto JUDGE;
9. else $i++$, $j++$;
10. goto JUDGE;
11. end if
12. else $p = p \rightarrow \text{next}[j]$
13. if $\text{End} = P[j]$
14. $j = \text{next}[j]$;
15. goto JUDGE;
16. else $i = i + 2$;
17. $j = 0$;
18. goto JUDGE;
19. end

该算法的优点是: 当目标串与源串匹配时, 如果源串出现不匹配的字符时, 该目标串直接与源串的下一个字符开始匹配. 图1是本文提出的基于二进制编码后的3次匹配过程演示. 其中 T 是源串, 3次匹配的源串都不变. 而 P_1, P_2, P_3 各表示第1次, 第2次, 第3次匹配的目标串; 这里 T^*, P_1^*, P_2^*, P_3^* 分别表示二进制编码后的串.

2.2 复杂度分析

本文将验证如果使用任何亚线性算法对数字字

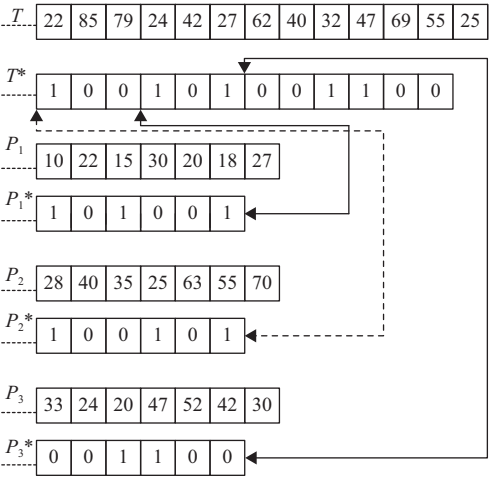


图1 3次匹配过程演示

Fig.1 A three-pass demonstration of the process

符进行二进制转换,那么该处理方法在任何情况下其平均时间复杂度是亚线性的.

假设 P 、 T 数字字符都满足这样的条件: P 中的数值与 T 中数值都是相互独立的、均匀的、离散的整数. P^* 、 T^* 是转换后的集合, c 表示集合中整数出现的次数,由于在 P^* 、 T^* 中有 c^2 个整数对跟 c 个等式,那么在 P^* 、 T^* 中整数出现在任意一个位置的概率就为

$$g = \frac{\frac{c^2}{2} - \frac{c}{2}}{c^2} = \frac{c-1}{2c}, \tag{5}$$

因此,一个数值字符匹配的概率 h 为

$$h = g^2 + (1-g)^2 = 2g(g-1) + 1 = \frac{1}{2} + \frac{1}{2c^2}. \tag{6}$$

在 $P^* = b_1b_2 \cdots b_{m-1}$ 及 T^* 中内部相邻的位置其实并不是独立的,考虑段落的宽松匹配 $P^{**} = b_1b_3b_5 \cdots b_s$, P^{**} 中包含了 P^* 中的0、1与其他任何的字符,其中 $s = 2\lfloor m/2 \rfloor - 1$.那么 P^{**} 与 T^* 匹配的概率要小于 $h^{(m-1)/2}$,随着 m 的逐渐增大,概率会趋向为0,当 $c=2$ 时, P^* 匹配的概率同样小于 P^{**} 匹配的概率.表明本文中提出的处理方法在任何情况,如果转换成二进制形式的时间复杂度为线性的,则总的时间复杂度是线性的.本文也考虑了最坏的情况,如果 P^* 为 1^{m-1} 且 T^* 为 1^{n-1} ,则总的时间复杂度则为 $O(nm)$.

3 实验与结果分析

本文测试了4种保序匹配算法.有两种基于BNDM算法^[1],分别是SBNDM2、SBNDM4.在BNDM算法中,每一个对齐窗口的处理方式都跟Boyer-Moore算法一样是从右向左进行处理的.第3种算法为Fast Shift-Or (FSO)^[10].这里之所以选择该

算法是因为这种算法处理二进制文本速度非常快.第4种算法为KMP算法^[8, 10-12];SBNDM2与SBNDM4算法时间复杂度是亚线性的,FSO与KMP算法时间复杂度是线性的.为了提升过滤效率本文选用了基于误配q-gram过滤器^[13].这里把长度为 x 的q-gram和模式长度为 m 的 P 定义成: $l_x = \max\{i|u_p[i-q+1..i] = u_x, q-1 \leq i < m-1\}$.这里, l_x 表示在 P 中上次的位置.通过引理1知,若 $u_p[i-q+1..i] \neq u_x$,则 $p[i-q+1..i] \neq x$,因此不会漏掉任何一个匹配位置,所以匹配移动的窗口 D 定义为: $D[f(x)] = \min(m-q+1, m-l_x-1)$, $f(x)$ 使用前缀表达式来表示为

$$f(x) = \sum_{k=0}^{q-1} u_x[k] * k!. \tag{7}$$

注意:若前缀表达式从0至 $q!-1$ 都映射,则 D 需要 $O(q!)$ 空间.实验环境配置采用的是2.70 GHz的CPU,16 GB的内存,操作系统为Ubuntu 12.10版本.

实验数据是随机产生的,数据的范围是0至 2^{30} 的整数,数据集长度为1 000 000.在总数据集中随机选取匹配库长度为 $n=1\,000$.待匹配的长度取4种情况: m 分别为5、10、15、20.表1为实验测试的4种算法的平均执行时间的对比,本文中分别对4种算法进行了80次实验,表1中的SBNDM2、SBNDM4、FSO、KMP是未改进的算法;SBNDM2(1)、SBNDM4(1)、FSO(1)、KMP(1)算法是基于二进制编码改进后的的算法.

表1 4种算法的性能测试对比

Tab.1 Comparison of performance tests for four algorithms

数据	算法	m值(匹配时间/s)			
		5	10	15	20
随机产生的数据	SBNDM2	3.91	1.90	1.16	1.26
	SBNDM2(1)	3.80	1.77	1.02	1.10
	SBNDM4	3.91	1.78	1.15	0.86
	SBNDM4(1)	3.80	1.66	1.02	0.68
	FSO	5.35	1.65	1.69	1.66
	FSO(1)	5.30	1.51	1.49	1.42
	KMP	6.85	6.62	6.03	4.92
	KMP(1)	6.74	6.51	5.88	4.71

表1中的实验结果可以用来做一个性能比较,不难发现以下规律:(1)当 $m=5$ 时,SBNDM2算法与SBNDM4算法匹配速度是比较快的,KMP算法匹配速度是最慢的;(2) $m=10$ 时,FSO算法匹配速度是最快的,KMP算法匹配速度是最慢的;(3) $m=15$ 或20时,SBNDM4算法匹配速度是最快的,KMP算法匹

配速度是最慢的;(4) 特别的是SBNDM2算法与SBNDM4算法两者的匹配速度是很接近的;(5) 随着 m 取值的增加,匹配时间有减少的趋势;(6) 所取的4种值不管为何值时,KMP算法匹配速度总是最慢的;(7) 改进后的算法提高了匹配效率。

4 结束语

本文针对数值保序匹配问题提出了一种基于二进制编码处理方法。在研究中使用了SBNDM2、SBNDM4、FSO、KMP 4种算法来实现并且对提出的新的处理方式进行了实验测试对比。理论分析和实验结果表明:基于二进制编码^[14]的处理方法,能够进行正确有效的匹配,并且具有较低的空间和时间复杂度;提高了数字字符匹配的计算效率;有效降低了数字字符匹配的时间消耗;有较大的理论与应用价值。当然本文中的方法在预处理上会有时间消耗,但其时间复杂度是线性的,且总体的匹配计算时间会大大减少,如何使总体匹配时间进一步的减少将是后续研究的重点。

参考文献:

- [1] NAVARRO G, RAFFINOT M. Flexible Pattern Matching in String Practical On-line Search Algorithms for Texts and Biological Sequences[M]. Cambridge: Cambridge University Press, 2002. 23-28.
- [2] FARO S, LECROQ T. The exact online string matching problem: A review of the most recent results [J]. ACM Computing Surveys, 2013, 45(2): 13-13.
- [3] KNUTH D E, MORRIS J H, PRATT V R. Fast pattern matching in string [J]. SIAM Journal on Computing, 1977, 20(6): 323-350.
- [4] BOYER R S, MOORE J S. A fast string searching algorithm [J]. Communications of the ACM, 1977, 20(10): 762-772.
- [5] 苏德福, 钟诚. 计算机算法设计与分析[M]. 2版. 北京: 清华大学出版社, 2001. 57-59.
- [6] DANIEL M S. Very fast substring search algorithm [J]. Communications of the ACM, 1990, 33(8): 132-142.
- [7] AHMED M, KAYKOBAD M, CHOWDHURY R A. A new string matching algorithm [J]. Computer Mathematics, 2003, 80(7): 825-834.
- [8] 赵森严, 黄伟, 李阳铭. 一种改进的KMP入侵检测的模式匹配算法[J]. 井冈山大学学报(自然科学版), 2013, 34(1): 55-58.
- [9] ZHAO S Y, HUANG W, LI Y M. Improved model matching algorithm for KMP Intrusion detection [J]. Journal of Jinggangshan University: Natural Science, 2013, 34(1): 55-58.
- [10] AMIR A, FARACH M. Efficient 2-dimensional approximate matching of halfrectangular figures [J]. Information and Computation, 2010, 118(1): 1-11.
- [11] 欧鬼, 吴纯青. 几种字符串匹配算法的分析和比较[J]. 微处理机, 2007, 28(4): 59-62.
- [12] OU W, WU C Q. Analysis and comparison of several string matching algorithms [J]. Microprocessors, 2007, 28(4): 59-62.
- [13] 杨俊丽, 吕晓燕, 满晰. 基于改进的KMP算法的词频统计[J]. 微计算机信息, 2010, 26(9): 161-162.
- [14] YANG J L, LYU X Y, MAN X. Based on improved KMP algorithm for word frequency statistics [J]. Microcomputer Information, 2010, 26(9): 161-162.
- [15] 李莉, 江育娥, 林劼, 等. 基于KMP算法的改进算法KMPP[J]. 计算机工程与应用, 2016, 52(8): 33-37.
- [16] LI L, JIANG Y E, LIN J, *et al.* Improved algorithm KMPP based on KMP [J]. Computer Engineering and Applications, 2016, 52(8): 33-37.
- [17] CLIFFORD R, ILIOPOULOS C S. Approximate string matching for music analysis [J]. Soft Computing, 2004, 8(9): 597-603.
- [18] 陈文伟, 赵侠, 黄金才. 进化创新的绕行变换[J]. 广东工业大学学报, 2017, 34(1): 1-5.
- [19] CHEN W W, ZHAO X, HUANG J C. Modeling transformation of evolutionary innovation [J]. Journal of Guangdong University of Technology, 2017, 34(1): 1-5.