DOI :10.13245/j.hust.2006.10.013
第34卷 第10期 华中科技大学学报(自然科学版) Vol. 34 No. 10
2006年 10月 J. Huazhong Univ. of Sci. & Tech. (Nature Science Edition) Oct. 2006

一种改进的KMP高效模式匹配算法

鲁宏伟 魏 凯 孔华锋

(华中科技大学 计算机科学与技术学院, 湖北 武汉 430074)

摘要: 针对KMP算法存在着主串与模式串中多个相同字符重复比较的缺陷, 在KMP算法的基础上, 给出了一种新的模式匹配算法, 该算法不像KMP算法那样向左滑动模式串的指针, 而是每次比较字符不匹配时, 根据模式串当前字符的特征值 k, 使主串的指针向前跳跃 k 个值, 且使模式串的指针置于起始位置, 开始新一轮的匹配, 加快了主串的匹配速度. 理论分析和试验证明, 该算法需要的比较次数比KMP算法减少将近一半.

关 键 词: 模式匹配; 算法; 模式串; 主串; 时间复杂度

中图分类号: TP391.4 文献标识码: A 文章编号: 1671-4512(2006)10-0041-03

An improved high-effective KMP pattern matching algorithm

Lu Hongwei Wei Kai Kong Huafeng

(College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: The comparison between text strings and pattern strings was carried out repeatedly in KMP pattern matching algorithm. A new arithmetic for pattern matching is proposed, based on the KMP algorithm. It is deferent from the KMP algorithm that when each matching is not accomplished, the pointer of text strings is shifted forward by k units according to eigenvalue k of the current characters in pattern strings and the pointer for pattern strings is placed in the starting position for new comparison, in this new algorithm. The speed of matching can be improved. Theoretical and experimental results show that the principal advantage over KMP pattern matching algorithm is nearly a two-fold reduction in the required times of comparison.

Key words: pattern matching; algorithm; pattern string; text string; time complexity

由于朴素的模式匹配算法复杂度较高, 人们提出了一些复杂度较低而效率更高的算法, 如BF算法、BM算法、RF算法和KMP算法. 其中效率最高的是KMP模式匹配算法[1], 它充分利用了以前匹配不成功的信息而大大提高了匹配的效率, 其算法复杂度为 $T(n)=O(m+n)$. 然而在KMP算法中, 会出现主串中的一个字符与模式中的多个相同字符重复地作不必要比较的情形, 这种情况有时使算法的效率降低许多[2]. 本文针对KMP算法的这个缺陷, 设计了一种新的算法, 减少比较次数, 从而提高匹配效率.

1 KMP模式匹配算法

所谓模式匹配是在给定主串的字符串 S 中寻找与给定的另一个称为模式串的字符串 T: $t_0t_1\cdots t_{m-1}$ $(n>m)$完全匹配的所有子串的位置. 若模式串 T 与主串 S 中的某子串匹配, 则匹配成功, 返回该子串在主串中的位置; 否则返回0, 匹配失败. 为以下讨论方便, 定义 $a_1a_2\cdots a_m=b_1b_2\cdots b_m$, 当且仅当 $a_i=b_i$, $i=1, 2, \cdots, m$. 匹配结果用数学公式表达为 $F=\{i \mid 0\leqslant i\leqslant n-m, s_is_{i+1}\cdots s_{i+m-1}=t_0t_1\cdots t_{m-1}\}$, 如果匹配成功, 就返回模式串的第一

收稿日期: 2005-09-22.

作者简介: 鲁宏伟(1964-), 男, 教授; 武汉, 华中科技大学计算机科学与技术学院(430074).

E-mail: luhw@hust.edu.cn

个字符在主串中的位置 i. 由于模式串可能在主串中多次出现, 因此 F 是一个非负整数集, 包含了模式串在主串中出现的所有位置. 如 $S=$"sdjklpaotheipaomil", $T=$"pao", 则 $F=\{5, 12\}$.

KMP 算法中, 先给模式串 T 的每个字符赋予一个值, 用 next_j $(j=0, 1, \cdots, m-1)$ 来表示, 定义如下[1]:

$$\mathrm{next}_j=\begin{cases}-1 & (j=0),\\ \max\limits_{k\in A_j}\{k\} & (A_j=\{k \mid 0<k<j,\\ \quad t_0t_1\cdots t_{k-1}=t_{j-k}t_{j-k+1}\cdots t_{j-1}\}\neq\varnothing),\\ 0 & (\text{其他}).\end{cases} \tag{1}$$

假设模式串 $T=$"sdgassda", 根据式(1)计算 next_j 值如表 1 所示.

表 1 KMP 算法求 next_j 值的结果

j	t_j	next_j	j	t_j	next_j
0	s	−1	4	s	0
1	d	0	5	s	1
2	f	0	6	d	1
3	a	0	7	a	2

KMP 模式匹配算法的基本思想如下:

A. 按照式(1)求得模式串中每个字符的 next_j 值.

B. 进行模式匹配. 假设 i 和 j 分别为指示主串和模式串中正在比较的字符的当前位置, 并对 i 和 j 赋初值 0. 在匹配的过程中, 若 $s_i=t_j$, 则 i 和 j 分别增加 1, 继续进行比较; 否则, i 不变, 而 j 退回到 next_j 的位置进行新一轮的比较, 如此递推下去, 直到出现下列两种情况:

a. 当 j 退回到某个值 next_j 值时, 匹配成功, 则 i 和 j 分别增加 1, 继续匹配;

b. 当 j 退回到值为 0 时, 即 $\mathrm{next}_j=-1$, 说明主串的当前字符匹配失败, 这时将主串向右滑动一个位置, 即从 $i+1$ 处重新开始新一轮的匹配, 此时 $j=0$.

2 本文算法

定义 1 定义集合

$$B_j=\{r \mid 0<r<j, t_rt_{r+1}\cdots t_{j-1}=t_0t_1\cdots t_{j-1-r}\}\ (j=0, 1, \cdots, m-1),$$

与 KMP 算法类似, 定义模式串

$$K_j=\begin{cases}1 & (B_j=\varnothing);\\ \min\limits_{r\in B_j}r & (B_j\neq\varnothing),\end{cases} \tag{2}$$

称 K_j 为模式串 T 中字符 t_j 的特征值.

例如, 当 $T=$"sdsdagas"时, 根据式(2)计算得到的 K_j 值如表 2 所示.

表 2 本文算法求 K_j 值的结果

j	t_j	K_j	j	t_j	K_j
0	s	1	4	a	2
1	d	1	5	g	1
2	s	1	6	a	1
3	d	2	7	s	1

定义 2 主串 S 与模式串 T 的一次匹配失败是指 $s_ps_{p+1}\cdots s_{i-1}=t_0t_1\cdots t_{j-1}$, 但 $s_i\neq s_j$, 其中 p 为主串 S 中该次匹配的起始位置.

定理 1 若主串 S 与模式串 T 的一次匹配失败, 主串与模式串当前的位置分别为 i 和 j, 且主串对应模式串首字符的位置为 p $(p<i)$, 则有 $p=i-j$.

根据定义 2, 该结论显然.

定理 2 若主串 S 与模式串 T 的一次匹配失败, 主串与模式串当前的位置分别为 i 和 j, 且主串对应模式串首字符的位置为 p $(p<i)$, 则不存在整数 q $(0<p<q<p+K_j)$, 使得 $s_qs_{q+1}\cdots s_{i-1}=t_0t_1\cdots t_{i-q-1}$.

证明 假设存在这样的 q, 则由定义 2, $s_qs_{q+1}\cdots s_{i-1}=t_{q-(i-j)}\cdots t_{j-1}$, 又由定理 1, $t_{q-p}\cdots t_{j-1}=t_{q-(i-j)}\cdots t_{j-1}=s_qs_{q+1}\cdots s_{i-1}=t_0t_1\cdots t_{i-q-1}=t_0t_1\cdots t_{j-1-(q-p)}$. 由此及定义 1, 可知 $K_j\leqslant q-p$, 即 $q\geqslant K_j+p$, 这与 $q<p+K_j$ 矛盾.

定理 2 说明, 当一次匹配失败时, 假设主串对应模式串首字符的位置为 p, 下一轮的比较从主串 S 的第 $p+K_j$ 个位置开始, 与模式串 T 第 1 个字符开始比较, 才能使主串 S 与模式串 T 比较的字符走到 $i-1$ 的位置仍旧能匹配, 否则, 这次匹配就没有意义.

由定理 2 给出相应的匹配算法如下:

a. 根据式(2)计算 K_j $(j=0, 1, \cdots, m-1)$.

b. 进行模式匹配, 假设 i 和 j 分别为指示主串和模式串中正在比较的字符的当前位置, 并对 i 和 j 赋初值 0, $p=0$.

c. 若 $s_i=t_j$, 则 $p=i$, i 和 j 分别增加 1, 跳转到步骤 **e**; 否则执行下一步骤.

d. $i\leftarrow p+K_j$, 而 $j\leftarrow 0$, 回到上一步, 继续进行比较.

e. 若 $s_i=t_j$, 则 i 和 j 分别增加 1, 重复本步骤; 否则执行上一步骤.

下面给出一个根据上述算法进行匹配的实例. 假设 $S=$"fdsdsbsdsdagaslk", $T=$"sdsdagas", 根据式(2)计算 K_j 值, 匹配过程如下:

第一轮

↓i=0

主串　f d s d s b s d s d a g a s l k

模式串　s d

↑j=0, K_j=1.

第二轮

↓i=1

主串　f d s d s b s d s d a g a s l k

模式串　　s d

↑j=0, K_j=1.

第三轮

↓i=5

主串　f d s d s b s d s d a g a s l k

模式串　　s d s d

↑j=3, K_j=2.

第四轮

↓i=5

主串　f d s d s b s d s d a g a s l k

模式串　　s d s d a g a s

↑j=1, K_j=1.

第五轮

↓i=5

主串　f d s d s b s d s d a g a s l k

模式串　　s d s d a g a s

↑j=0, K_j=1.

第六轮

↓i=13

主串　f d s f s b s d g a s s d a l　k

模式串　　s d g a s s d a

↑j=7

匹配成功.

3　算法时间复杂度的比较以及试验结果

算法时间复杂度可分两部分进行讨论，一是模式匹配算法中 S 与 T 的比较次数，二是根据模式串来计算 K_j. 因为 KMP 算法中计算 next$_j$ 与本文算法计算 K_j 涉及的运算量相同，所以这两个算法的复杂度主要区别在循环比较的次数上. 设 $P(m, n)$和$W(m, n)$分别为 KMP 算法、本文算法中 S 与 T 的平均比较次数，则：

a. 模式串中无重复出现子串的情况下，K_j 全为 1，可知 $W_1(m, n) \leqslant m+n$，此时对应的 next$_j$={－1, 0, 0…}，$P_1(m, n) \leqslant m+n$，两种算法在不同情况下互有优劣.

b. 一般情况下，假设上一次匹配不成功时主串的指针在 i 处，模式串在 j 处，亦即上一轮比较次数为 $j+1$ 次，可以知道 $s_i \neq t_j$，但

$$s_{i-j}s_{i-j+1}\cdots s_{i-1} = t_0\cdots t_{j-1}. \tag{3}$$

假设新的一轮主串指针起始位置在 r 处，则由定理 2，$r=i-j+K_j$. 假设新的一轮有 p 次单个字符匹配成功，则有 $s_{r+p} \neq t_p$，但

$$s_r's_{r+1}\cdots s_{r+p-1} = t_0\cdots t_{p-1}. \tag{4}$$

令 $c=K_j$，则由 K_j 的定义可知

$$t_0t_1\cdots t_{c-1} = t_{j-c}\cdots t_{j-1}. \tag{5}$$

由式(3)～(5)可以得到此时 $s_rs_{r+1}\cdots s_{i-1}$ 进行了两次比较，因此比较次数比情况 **a** 时多出 $i-1-r=j-K_j$ 次.

同样在 KMP 算法中，进行新的一轮匹配时，主串指针在 $r'=i$ 处，模式串指针在 next$_j$ 处，可知相对于 **a** 的情况下重复比较了 $j-$next$_j$ 次.

因 $K_j \geqslant$ next$_j$，故有

$$j-K_j \leqslant j-\text{next}_j; \tag{6}$$

又由 **a** 的结果可得

$$\begin{cases} W(m, n) = W_1(m, n)+\sum(j-K_j), \\ P(m, n) = P_1(m, n)+\sum(j-\text{next}_j). \end{cases} \tag{7}$$

由式(6)和(7)可知，本文算法和 KMP 算法[3] 在时间复杂度上都在 $O(m+n)$级别，但本文算法的比较次数要比 KMP 少，特别在模式串中重复子串较少情况下优于 KMP 算法. 大量数据测试结果(表 3)也证明，对不同长度(这里的长度是指字符个数)的主串和模式串，本文算法的比较次数约为 KMP 算法的 50 %.

表 3　算法比较次数对比表

主串长度	模式串长度	比较次数	
		KMP 算法	本文算法
765	5	1 391	727
1 733	8	3 412	1 710
1 731	6	3 462	1 761
282	8	466	244

参　考　文　献

[1] 严蔚敏，吴伟民. 数据结构[M]. 2版. 北京：清华大学出版社，1998.

[2] 卢开澄. 计算机算法导引——设计与分析[M]. 北京：清华大学出版社，1996.

[3] Navarro G, Fredriksson K. Average complexity of exact and approximate multiple string matching[J]. Theoretical Computer Science, 2004, 321(2-3): 283-290.