文章编号:1001 - 9081(2006)02 - 0318 - 02

一种改进的 BM 模式匹配算法

杨薇薇,廖 翔
(华中科技大学 计算机科学与技术学院, 湖北 武汉 430074)
(hi_liaoxiang@163.com)

摘 要:首先介绍了目前最常用的 BM 模式匹配算法,以及其改进算法 Boyer-Moore-Horspool (BMH)算法,在此基础上提出了另一种改进的 BM 算法,该算法减少了匹配次数,有效的加快模式匹配的速度。

关键词:模式匹配;BM 算法; Boyer-Moore-Horspool 算法
中图分类号: TP309 **文献标识码**:A

Improved pattern matching algorithm of BM

YANG Wei-wei, LIAO Xiang
(*College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan Hubei* 430074, *China*)

Abstract: Pattern matching is a very important part of computer basic technology, It is widely used in engineering application. Firstly, the BM and BMH, which were most fashionable pattern matching algorithms at present were introduced. On the basis of them, an improved pattern matching algorithm was proposed, which is more efficient and has better performance.

Key words: pattern matching; BM algorithm; Boyer-Moore-Horspool algorithm

0 引言

模式匹配问题如下:要在一个文本 $T[1:n]$ 中查找某个特定的子串,使得这个子串与模式 $P[1:m]$ 相等。如果文本 T 中找到等于模式 P 的子串,则称匹配成功,否则称匹配失败。

求解上面提出的模式匹配问题,典型的算法有两种:BF 算法和 KMP 算法。BF 算法非常简单,但它可能产生不必要的回溯,减慢了模式匹配的速度;KMP 算法是由 BF 改进后不产生回溯的一种算法,可以提高匹配算法的效率,但相对比较复杂。在 KMP 算法的启发下,文献[1]提出了一种 BM 算法来求解模式匹配问题。

1 BM 算法

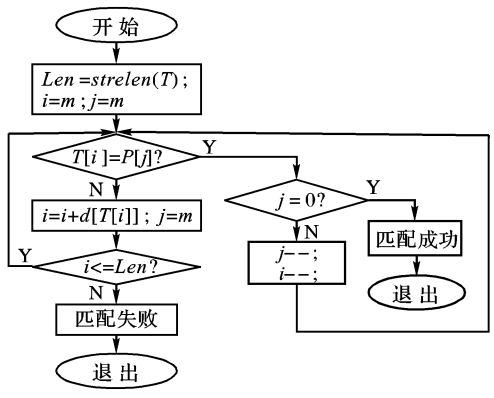

图 1 原 BM 算法

BM 算法的关键是根据给定的模式 $P = P1P2\cdots Pm$,定义一个函数 $d: x \rightarrow \{1,2,\cdots,m\}$,这里 $x \in \Phi$(Φ 是字符集)。函数 d 给出了正文中可能出现的字符在模式中的位置。

$$d(x) = \begin{cases} m & x \text{ 不在 } P \text{ 中出现或 } x \text{ 只出现在 } P \text{ 的尾部} \\ m - j & \text{期于情况,其中 } j = \max\{j : P_j = x, 1 \leqslant j \leqslant m-1\} \end{cases}$$

主要思想:假设在执行文本中自位置 i 起与模式的从右至左的匹配检查中,一旦发现不匹配(不管在什么位置,假设在 i' 处发生不匹配),则去执行由 P_m 与 $T_{i'+d(x)}$ 起始的从右至左的新一轮匹配检查,这里 x 为字符 $T_{i'}$。算法的流程如图 1 所示。

2 Boyer-Moore-Horspool(BMH)算法[3]

表 1 BMH 算法的匹配过程

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
文本 T	s	u	b	s	t	r	i	n	g	s	e	a	r	c	h
第 1 次	s	e	a	r	c	h									
第 2 次			s	e	a	r	c	h							
第 3 次									s	e	a	r	c	h	
第 4 次										s	e	a	r	c	h

注释:第 1 次,$i = 6$,$d[r] = 2$,向右滑 2 个字符。第 2 次,$i = 8$,$d[n] = 6$,向右滑 6 个字符。第 3 次,$i = 14$,$d[c] = 1$,向右滑 1 个字符。第 4 次匹配成功。

1980 年,Horspool 提出了对 BM 算法的一种改进算法——Boyer-Moore-Horspool 算法。它首先比较文本指针所指字符和模式的最后一个字符,如相等再比较其余 $m - 1$ 个字符。无论文本中哪个字符造成了匹配失败,都将由文本中和模式最后一个位置对应的字符来启发模式向右的滑动。即文本自位置 i 起与模式的从右至左的匹配检查中,一旦发现不匹配,就将 i 重新赋值为 $end + d[T[end]]$。(end 是一个中间变

收稿日期:2005 - 08 - 21;**修订日期**:2005 - 10 - 27

作者简介:杨薇薇(1945 -),女,湖北武汉人,副教授,主要研究方向:网络与信息安全、网络系统结构、图形图象处理; 廖翔(1981 -),男,湖北武汉人,硕士研究生,主要研究方向:网络与信息安全、网络系统结构.

量,记录了文本中每次从右至左匹配的起始位置)例如:文本 T 为"substringsearch",模式 P 为"search"。则用 BMH 算法进行匹配的过程如表 1 所示。

3　一种改进的 BM 算法

在完全随机的情况下,文本中某位置出现任何一个字符 x ($x \in$ 字符集 Φ) 的概率都为 q。假设在文本和模式的匹配检查中,由文本指针所指某位置出现的字符 x 启发模式向右的滑动,滑动位数为 $dist(x)$,而由此位置字符所启发的模式向右滑动位数的数学期望(均值)为 E。

BM 算法:

$$E_1 = \sum_{x \in \Phi} dist_1(x) * q, (x \text{ 是指针 } i' \text{ 所指字符}) \quad (1)$$

BMH 算法:

$$E_2 = \sum_{x \in \Phi} dist_2(x) * q, (x \text{ 是指针 } end \text{ 所指字符}) \quad (2)$$

其中:$dist_1(x) = (i' + d(x)) - end$ (x 是指针 i' 所指字符) (3)

$dist_2(x) = (end + d(x)) - end$ (x 是指针 end 所指字符) (4)

又因为 $i' \leqslant end$, 显然对于同样的字符 x, $dist_1(x) \leqslant dist_2(x)$,所以 $E_1 \leqslant E_2$,即 BMH 算法可以加大模式滑动的幅度,比 BM 算法具有更加好的性能。

通过文本中和模式最后一个位置对应字符的下一个字符可以启发模式向右滑动的位数。特别是这个字符不在模式中出现时,则可以将模式整体滑过该字符,这样就把滑动位数的最大值由 m 增大到了 $m+1$。

通过上述分析,我们将 BM 算法做进一步的改进:由文本中和模式最后一个位置对应的字符 $T[end]$,以及它的下一个字符 $T[end+1]$ 共同启发模式向右的滑动。其效果相当于在每次匹配失败后,把模式向右滑动这两个字符所启发位数中最大的位数。

改进算法的滑动位数:

$$dist_{23}(x_2, x_3) = \max(dist_2(x_2), dist_3(x_3)) \quad (5)$$

改进算法滑动位数的数学期望(其中 x_2, x_3 分别是指针 end, $end+1$ 所指字符):

$$E_{23} = \sum_{x_2 \in \Phi} \sum_{x_3 \in \Phi} \max(dist_2(x_2), dist_3(x_3)) * q^2 \quad (6)$$

显然 $E_{23} \geqslant \sum_{x_2 \in \Phi} \sum_{x_3 \in \Phi} dist_2(x_2) * q^2 = \sum_{x_2 \in \Phi} dist_2(x_2) * q$($q$ 是字符集 Φ 的基数的倒数)

即 $E_{23} \geqslant E_2 \geqslant E_1$,所以改进算法进一步加大了模式滑动的幅度,减少了匹配次数。

有一点值得注意,当 $T[end+1] \neq P_m$ 字符时,由 $T[end+1]$ 字符所启发的滑动位数 $dist_3(x) = (end + 1 + d(x)) - end$;当 $T[end+1] = P_m$ 字符时,$dist_3(x) = 1$,然而 $dist_2(x) \geqslant 1$。因此:

$$dist_{23}(x_2, x_3) = \max(dist_2(x_2), dist_3(x_3)) = \begin{cases} dist_2(x_2) & x_3 = P_m \\ \max(dist_2(x_2), dist_3(x_3)) & x_3 \neq P_m \end{cases}$$

这样就可以直接用函数 d 来判断 $dist_2(x_2)$, $dist_3(x_3)$ 中的较大者。改进算法的流程如图 2 所示。

例如:同样的文本 T:"substringsearch",模式 P:"search"。则用改进算法进行匹配的过程如表 2 所示。

表 2　改进算法的匹配过程

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
文本 T	s	u	b	s	t	r	i	n	g	s	e	a	r	c	h
第 1 次	s	e	a	r	c	h									
第 2 次								s	e	a	r	c	h		
第 3 次										s	e	a	r	c	h

注释:第 1 次,$i=6$, $d[r]=2 < d[i]=6$,用字符 $'i'$ 启发,向右滑 7 个字符。第 2 次,$i=13$, $d[r]=2 > d[c]=1$,用用字符 $'r'$ 启发,向右滑 2 个字符。第 3 次匹配成功。

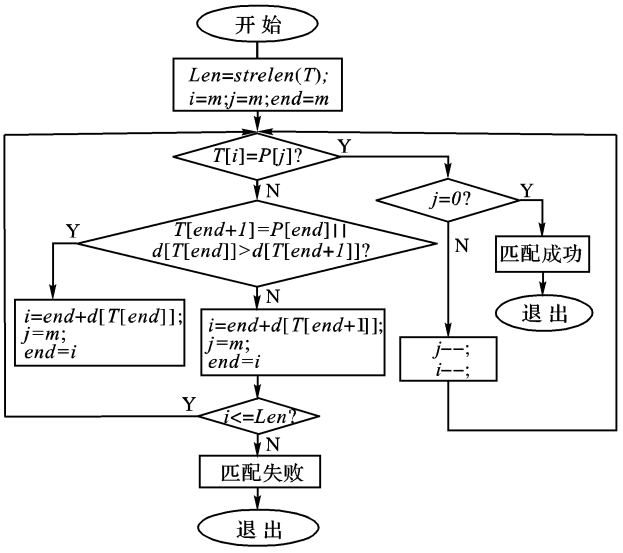

图 2　改进后的 BM 模式匹配算法

4　实验结果

测试 1　分别用不同的文本和模式,对原 BM 算法、BMH 算法、改进算法进行匹配次数的测试,测试结果如表 3 所示。

表 3　匹配次数比较

测试文本 T	模式串 P	文件大小 (KB)	原 BM 算法匹配次数	BMH 算法匹配次数	改进算法匹配次数
RFC1866	symbolically	135KB	12112	12081	10355
RFC2616	amounts	394KB	56698	56015	45581
RFC2910	attributes-natural-language	93.2KB	3378	3371	2816

测试 2　选用 2.18M 的纯英语文本,分别用长度为 2、5、8、10、20 五种不同模式对这几个算法进行执行时间的测试,测试结果如表 4 所示。

表 4　执行时间比较(时间:ms)

算法	长度				
	2	5	8	10	20
BM 算法	72.893	39.216	30.107	25.451	19.655
BMH 算法	61.510	36.191	29.207	24.237	18.884
改进算法	55.311	34.205	28.153	23.411	18.101

通过以上测试,可以看出本文提出的改进算法有效减少了匹配的次数,提高了模式匹配的速度。同时可以看出,在文本长度固定时,算法的执行时间与需要匹配的模式长度成反比。

参考文献:

[1] 李洋, 王康, 谢萍. BM 模式匹配改进算法[J]. 计算机应用研究, 2004, 21(4): 58 - 59.

[2] BOYER RS, MOORE JS. A fast string searching algorithm [J]. Communications of the ACM, 1977, 20(10): 762 - 772.

[3] HORSPOOL RN. Practical fast searching in strings[J]. Software—Practice and Experience, 1980, 10(6): 501 - 506.