A New Fast Patter Matching Algorithm

XIAO Shi-song, ZHENG Chuan-fen, ZHOU Zhang-lan, GUO Jun-ying

(Department of Computer Science, Huazhong Normal University, Wuhan 430079 China)

E-mail: cfzheng@126.com

Abstract: The pattern matching technique is a significant operation in computer science. Though the analysis of the commonly used single pattern matching algorithm—BM and of the existing improved algorithm—BMH, a new improved algorithm was proposed based on BMH. Considering the feature of the text string and the pattern string, this algorithm fully used the information which appeared in the text string but while not exist in the pattern string to expedite jumps toward right scope after the match defeat, effectively reducing the match times. The experiment result indicates that this algorithm can reduce the number of comparisons and that of the pattern string's skip, and raise the match efficiency greatly.

Keywords: pattern matching; BM algorithm; BMH algorithm; Improvement of BMH algorithms

一种新的快速模式匹配算法

肖诗松，郑传芬，周张兰,郭俊颖

（华中师范大学计算机科学系，武汉 430079）

E-mail:cfzheng@126.com

【摘要】模式匹配是计算机科学中的一种重要运算。通过对常用的单模式匹配算法 BM 和改进算法 BMH 的分析，提出了基于 BMH 算法的改进算法。该算法综合考虑文本串和模式串的特征，充分利用出现在文本串中而没有出现在模式串中的信息。当匹配失败时使模式串整体滑过文本串中的这些字符，加快了匹配失败后向右跳跃的幅度。实验结果证明该算法能够减少字符的比较次数和模式串的移动次数，有效地提高了匹配效率。

【关键词】模式匹配；BM 算法；BMH 算法；BMH 改进算法

1 引 言

模式匹配是计算机科学中的一种重要运算。在搜索引擎、网络入侵检测、信息检索以及 DNA 序列匹配等诸多方面都有重要的理论价值，因此对模式匹配算法进行优化和深入研究具有很重要的意义。以字符串搜索为例，模式匹配问题是指：在已知长度为 n 的文本字符串 T 中查找长度为 m(m≤n)的模式字符串 P。如果在 T 中查找到等于 P 的子串，则匹配成功，返回子串在 T 中的位置，否则匹配失败，返回 0。目前典型的单模式匹配算法有 KMP 算法、BM 算法以及基于 BM 算法的改进算法比如 BMH 算法。

KMP(knuth，morris, pratt)算法[1]基本思想是：当一轮匹配过程中出现失配时，不需回溯主串，而是充分利用已经得到的部分匹配的结果，将模式串向右“滑动”尽可能远的一段距离后，继续进行比较，从而提高模式匹配的效率。算法在最坏的情况下具有线性搜索时间，时间复杂度为 O(m+n)。

进而言之，由于 BM[2] (Boyer _Moore)算法使用了坏字符和好后缀启发式搜索，能够在搜索时跳过大部分文本，从而提高搜索效率，其速度往往比 KMP 算法快上 3 到 5 倍。BM 算法经过优化和改进，匹配效率更高，目前公认的效率最高的算法是 BMH 算法。经过分析和研究本文在介绍 BM 算法和 BMH(Boyer _Moore _ Horspool)算法的基础上，对 BMH 算法进行改进，加快匹配失败后向右跳跃的幅度，提高了模式匹配速度。

2 BM和BMH算法的介绍

2.1 BM 算法[2]

1977 年, Boyer 和 Moore 提出了著名的 BM 模式匹配算法。BM 算法的基本原理[3,4]如下：

1) 自右向左匹配：将模式串 P 和给定的文本 T 左端对齐，自右向左依次扫描比较，如果匹配则继续扫描，如果扫描能够到达最左端，则匹配成功；如果扫描过程中遇到不匹配字符，则取坏字符函数和好后缀函数值较大者为滑动距离，将模式串右移，然后重复上面的操作。

2) 坏字符函数：BM 算法的关键是对模式串 P 定义一个从字母到正整数的映射函数 d（x）。d:x→ {1, 2,. . . ,m}，d（x）称为滑动距离函数, 它给出了正文中

的任一字符 c 在模式串中的位置。d（x）函数的具体定义[5]如下:

$$d(x)=\begin{cases} m;\ c\neq P[j]\ (1\leqslant j\leqslant m-1),\text{即}x\text{在}P\text{中未出现}\\ \text{或出现在P的尾部}\\ m-j,\ j=\max\{j\mid P[j]=c,\ 1\leqslant j\leqslant m-1\},\text{其它情况}\end{cases}$$

在执行文本 T[i]与模式 P[m]的从右到左的匹配过程中，一旦发现不匹配假设在 T[j]，立即执行由 T[j+d[T[j]]与 P[m]的新一轮的匹配。

3）好后缀函数：若模式串 P 中后面的 k 位已经和文本串 T 中有一部分匹配成功（标记匹配成功的部分为子串 P1），此时检查模式串 P 中前边 m-k 位是否存在子串 P1，若存在则将模式串右移使之对齐。如果不存在，则检查在模式串 P 中前边 m-k 位是否存在子串 P1 的任意子串，如果存在，则右移模式串使其对齐。如果两者皆不存在，此时可直接将模式串向右移动 m 个长度。

BM 算法在不匹配的情况下使用了两种启发,即“坏字符”启发和“好后缀”启发,这两种启发均能产生最大值为 m 的移动距离，具有较高的匹配速率，因此在网络入侵检测系统中得到了广泛的应用。但是,由于“好后缀”启发的预处理和计算过程都比较复杂,并且在实际应用中，坏字符函数的应用的频率远高于好后缀规则，因此，BM 算法在此可进一步改进和优化。

2.2 BMH 算法[6,8]

Horspool 于 1980 年提出了改进与简化 BM 算法的 BMH 算法，其基本思想是通过文本串和模式串中字符的最后一个位置对应字符的字符来决定右移的字符个数。算法描述如下：

1）模式串与文本串左端对齐并且自右向左匹配；

2)首先比较文本指针所指字符和模式串的最后一个字符,如果匹配再继续比较。即当匹配开始时比较 T[end]和 P[m]从右至左依次检查匹配。一旦发现不匹配,若匹配失败发生在 P[j]≠T[i]，先根据 BM 算法计算出字符 d[T[end]]，然后将文本指针重新赋值为 end+d[T[end]](end 是一个中间变量,记录文本中每次从右至左开始比较的起始位置)，接着开始新一轮自 T[end+d[T[end]]]与 P[m]从右至左的匹配。依次循环，直到匹配成功或失败为止[7]。例如文本串 T 为“substringsearch”，模式串 P 为“search”，则用 BMH 算法进行匹配的过程如表 1 所示：

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

由表 1 可以看出,BMH 算法在字符匹配失败时仅仅采用了了坏字符函数。它简化了预处理过程，省去了计算“好尾缀”启发的移动距离，并省去了比较由坏字符函数和好后缀函数引起的移动距离的过程，并且该算法的最坏情况即为 BM 算法的情况，故相对于 BM 算法具有一定的优越性。

然而我们再次看下第二次匹配，其实这次匹配是没有必要的。因为 T[8]根本就没有在模式串中出现，所以说匹配是不会成功的。同样 T[9]也没有在模式串中出现，这时我们完全可以将模式串整体滑过 T[7]T[8]T[9]这三个字符，从字符 T[10]开始下一轮的匹配。由此可见 BMH 算法还是可以进一步改进的。

3 改进的BMH算法

通过对上述几种匹配算法的分析与研究和实际应用，本文提出了在 BMH 算法的基础上进一步改进的算法。算法基本思想描述如下：

当在文本串与模式串的比较过程中发现不匹配时,首先判断文本串的 T[end]的下一个字符 T[end+1]是否出现在模式串中，如果该字符没有出现则判断其下一个字符是否出现在模式串中，这时用整数 q 来记录 T[end]后没有出现在模式串中的字符的个数。直到字符 T[end+q+1]出现在模式串中,此时让 T[end+q+1]与模式串的第一个字符 P[1]对齐，重新开始新一轮的比较匹配。若字符 T[end+1]出现在模式串中则由 T[end]按坏字符函数来启发模式串向右滑动。依次循环，直到匹配成功或失败。当文本串中的字符没有在模式串中出现时，可将模式串整体滑过这些字符。这样把滑动距离的最大值由 m 增大到了 m+q(q≥0)，加快匹配失败后向右跳跃的幅度。其最坏情况下也就是

978-1-935068-01-3 © 2009 SciRes.

当 q=0 时即 BMH 算法的情况，故具相对于 BM 和 BMH 算法有一定的优越性。

改进的算法中 d(x)[8]的函数计算 d()如下：

```
void d(int m;int n;int d[])
{
 int i;
 for(i=0;i<n;i++)
 d[t[i]]=m;        /*构造滑动距离数组*/
 for(i=0;i<m-1;i++)
 d[p[i]]=m-i-1;
 return;
 }
```

与 BM 和 BMH 的滑动函数基本一致。在完成 d(x)函数之后，我们看下改进后的进行的字符匹配的 search()的具体实现：

```
void search(char p[];char t[];int m;int n;int d[])
{
 int i;j;end;q;
 i=m-1;
 q=0;
 while(i<n)
 {
j=m-1;
end=i; /* end 记录文本中每次从右至左开始比较的起始位置*/
while(j>=0&&p[j]==t[i])
{
    i--;
    j--;
 }       /*进行匹配*/
 if(j==-1)
 {
  printf("the position is %d\n",i+1); /*匹配成功，返回位置*/
 break;
 }
 else
 {
  while(d[t[end+q+1]]==m) /*判断 T[end]后的字符是否在模式串中出现*/
  q++; /*对没有出现在模式串的文本串字符进行计数*/
  if(q==0)
   i=end+d[t[end]]; /*T[end+1]出现在模式中，由 T[end]启发模式滑动 */
 else
 {
  i=end+q+m; /*模式串整体滑过没有出现在模式串中的连续 q 个字符*/
  q=0; /*将 q 清零，为下次计数做准备*/
  }
  printf(“failure\n”);  /*匹配失败，退出*/
  }
}
```

例如：同样的文本 T 为“substringserch”，模式串 P 为“search”，则用改进的 BMH 算法进行匹配的过程如表 2 所示：下面我们根据表 2 对改进算法的匹配过程做一个分析：

表2 改进的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

(1)第一次匹配是 T[6]与 P[6]进行比较，匹配失败。接着判断 T[6]是否出现在模式串中，没有出现，这时 q 开始计数。直到 T[10]出现在模式串中，此时 q＝3，可以计算出下次匹配的位置为 15。

(2)第二次匹配是 T[15]与 P[6]对齐重新进行比较，匹配，接着从右往左循环比较模式串中的所有字符与对应位置的文本串中的字符。直到 j=0 匹配成功，比较结束。

通过分析我们得知，上例中使用 BM 算法需要进行 9 次比较，而使用改进的 BMH 只需要进行 7 次比较，明显的减少了两次，并且模式串的移动次数也由 4 次减少到 2 次。由此可见，该算法有效地减少了字符重复比较的次数和模式串的移动次数，提高了算法的匹配效率。

4 算法比较实验结果

为了比较文中涉及的算法的效率，进行了如下测试：

测试　选用长度为 1000 的纯英文文本和不同长度的模式串对这几个算法进行了匹配次数比较的测试，实验环境为 IntelCPU1.8HZ，内存 512M，测试结果如表 3 所示：

表3 匹配次数的比较

算法	模式串的长度			
	2	6	15	30
BM 算法	59	35	111	93
BMH 算法	59	33	102	90
改进的算法	48	24	80	77
减少的比较次数	11	9	22	23

从表 3 的测试结果可以看出，在用这四种不同长度的模式串进行测试的过程中，改进的算法的匹配次数最少，特别是在模式串长度比较短时优势更为明显。

5 结束语

本文通过分析常用的单模式匹配 BM 和 BMH 算法，提出了一种新的快速模式匹配算法。该算法当每次匹配失败后跳过了文本串中无用的字符即根本没有在模式串中出现的字符，这样使得当文本串大量出现样本中根本没有的字符的情况下，加快了匹配失败后向后跳跃的幅度，减少了不必要的匹配，效率明显提高。经过比较可以看出，此改进的算法比 BM 和 BMH 算法比较次数有所减少，同时又没有增加算法的复杂度，对模式匹配有着较大的实用价值。

References (参考文献)

[1] Knut h D E, Morris J H, Pratt V R. Fast Patter Matching in String [J]. SIAM Journal on Computing, 1977, (6): 323-350.

[2] Boyer R S, Moore J S. A Fast String Searching Algorithm [J].Communications of ACM, 1977, 20 (10):762-772.

[3] Zhang Guoping, Xu Wendong. Improved matching algorithm of string pattern [J], *Computer Engineering and Design* [J], 2007, 28 (20): 4881-4884.
张国平，徐汶东。字符串模式匹配算法的改进[J]。计算机工程与设计，2007，28(20):4881-48884。

[4] Li Yang, Wang Kang, Xie Ping. The Improving Algorithm of BM [J]. *Application Research of Computer,* 2004, 21 (4): 58-59.
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。

[5] Wu Xihong. Anatomy of BM pattern matching algorithm [J], *Computer Engineering and Design,* 2007, 1 (1): 29-31.
巫喜红。BM模式匹配算法剖析[J]。计算机工程与设计, 2007，1(1)：29-31。

[6] Horspool N R. Practical Fast Searching in Strings [J] .Software Practice and Experience, 1980, 10 (6): 501-506.

[7] Yang Weiwei, Liao Xiang. Improved pattern matching algorithm of BM [J]. *Computer Applications,* 2006, 26(2): 317-319.
杨薇薇，廖翔。一种改进的BM模式匹配算法[J]。计算机应用, 2006,26(2)：317-319。

[8] Liu Shengfei, Zhang Yunquan. Improved Pattern Matching Algorithm of BMH [J].*Computer Science,* 2008, 35 (11): 164-165.
刘胜飞，张云泉。一种改进的BMH模式匹配算法[J]。计算机科学，2008,35(11):164-165。

978-1-935068-01-3 © 2009 SciRes.