文章编号:1671-4598(2018)05-0173-04 DOI:10.16526/j.cnki.11—4762/tp.2018.05.043 中图分类号:TP391 文献标识码:A

BF 模式匹配算法的改进

巫喜红[1]，文张斌[2]
(1. 嘉应学院 计算机学院，广东 梅州 514015；2. 广东暨通信息发展有限公司，广东 广州 510000)

摘要：文章分析经典的 BF 算法及其改进方法，根据字符串匹配的特点对 BF 算法提出了新的改进算法 I_BF 算法；I_BF 算法根据模式串的首字符与匹配窗口之间的距离来确定右移距离，从而进行快速地匹配，匹配方式是从左往右进行；为了测试 I_BF 算法的性能，在相同条件下，从匹配字符个数、匹配次数、所花时间三方面对 I_BF 算法进行实验；结果表明，由于 I_BF 算法能够很大程序地跳过坏字符，减少匹配次数和字符比较个数，节约匹配时间，从而有效地提高匹配速度。

关键词：BF 算法；I_BF 算法；首字符；模式匹配；改进

Improvement of BF Pattern Matching Algorithm

Wu Xihong[1], Wen Zhangbin[2]
(1. School of Computing, Jiaying University, Meizhou 514015, China;
2. Guangdong Ji Tong Information Development Co., LTD., Guangzhou 510000, China)

Abstract: In this paper, the classical BF algorithm and its improved method are analyzed. According to the characteristics of string matching, a new improved algorithm I_BF algorithm for BF algorithm is proposed. The I_BF algorithm determines the right distance according to the first character of the pattern string and the distance between the matching windows, so it fasts matching. The matching way of I_BF algorithm is from left to right. In order to test the performance of the I_BF algorithm, the I_BF algorithm is tested under the same conditions from three aspects: the number of characters, the number of matches and the time spent. The experimental results show the I_BF algorithm is more quickly and more efficient because it reduces greatly the number of matching and character comparison for maximizing to skip the bad characters.

Keywords: BF algorithm; I_BF algorithm; first character; pattern matching; optimization

0 引言

随着计算机网络时代的飞速发展，人们足不出户就可以做任何事情，比如网购、看新闻、发表网络小说等。网络的利民之处众所周知，但也时不时看到这样的新闻，网银密码给黑客窃取，某公司网站被黑客“逛”，顺手拿走内部资料……因此，网络的安全性关系到人们的切身利益。在网络安全领域中，具有网络安全性的入侵检测系统也越来越广泛地被应用到生活中，设计者也越来越关注入侵检测系统的关键技术——模式匹配算法。在前人不断的研究和积累中，对模式匹配算法及其改进已有丰富的成果，如典型的单模式算法有 Brute－Force（BF）算法[1-3]、Horspool 算法[4]、KR 算法[5]，文献［6－7］提到的多模式算法主要有 Aho－Corasick（AC）算法、ACBM 算法及文献［8］提到的 Wu_Mander（WM）算法。为了提高算法的性能，研究者在这些算法的基础上对其进行不断地改进。本文从模式串的首字符与模式串的相关特点着手，对 BF 算法进行改进，称改进后的 BF 算法为 Improved_BF 算法，后文简称 I_BF 算法。

收稿日期:2017－09－01； **修回日期**:2017－10－18。

基金项目:2013 年广东省科技计划项目(2013B040500010);2014 年度广东省科技前沿与关键技术创新专项(2014B010117002);2016 年广东省重点平台及科研项目(2016KTSCX129);2016 年嘉应学院自然科学重点项目(2016KJZ04)。

作者简介:巫喜红(1975－),女,广东梅州人,硕士,副教授,主要从事算法理论、软件工程方向的研究。

1 BF 算法

模式匹配的定义是：对于给定的文本串 $T=T_0T_1\cdots\cdots T_{n-1}$（$n$ 为文本串的长度）和模式串 $P=P_0P_1\cdots\cdots P_{m-1}$（$m$ 为模式串的长度），（n 远大于 m）（T 和 P 都建立在有限字符集上，大小为 σ），要求在主串 T 中寻找等于模式串 P 的子串，如果在 T 中存在等于 P 的子串，则称匹配成功，函数值返回为 P 中第一个字符相等的字符在主串 T 中的序号，否则称为匹配失败，函数值返回为 0，此过程就是模式匹配。在许多改进的算法中，当匹配成功时，函数值返回 P 中所有匹配的字符在 T 中的位置。

经典的单模式匹配算法 BF（Brute－Force）算法是朴素模式匹配算法[9]。BF 算法的思想就是将目标串 T 的第一个字符与模式串 P 的第一个字符进行匹配，若相等，则继续比较 T 的第二个字符和 P 的第二个字符；若不相等，则比较 T 的第二个字符和 P 的第一个字符，依次比较下去，直到得出最后的匹配结果。BF 算法是一种蛮力算法。

BF 算法易于理解，但由于要比较的字符多，模式匹配的时间代价主要用于比较字符，所以时间效率不高。

BF 算法最好情况是，第一次匹配即成功，模式串刚好与目标串的开头长度为 m 的字符串 $T_0T_1\cdots\cdots T_{m-1}$ 匹配，此时比较次数为模式串长度 m，时间复杂度为 $O(m)$。最坏情况下要进行 $m*(n-m+1)$ 次比较，时间复杂度为 $O(m*n)$[10]。

2 改进的 BF 算法——I_BF 算法

2.1 已有改进算法

由于 BF 算法从 T 中的第一个字符开始进行比较，没有考

虑已取得的部分匹配的情况，为了减少比较次数，提高匹配效率，文献［1］对 BF 算法进行了改进，新改进的算法称为 EBF 算法。EBF 算法中抽象模式串中的字符是从整个模式串中选出来的特殊的字符，它的内容体现了整个模式串的内容特征，其字符的位置贯穿整个模式串，相对于 BF 算法从头开始一一进行匹配，能加快字符匹配失败的速度，其匹配效率可能就是 BF 算法一一匹配时的几倍或更高。

虽然文献［1］中的 EBF 算法优于 BF 算法，但能否根据模式串本身的特点在文本串的位置，从其它角度对 BF 算法进行改进呢？

2.2 I_BF 算法描述

为方便描述改进的 BF 算法，在此对一些变量说明如下：

字符指针变量 text：存放文本串

字符指针变量 pattern；存放模式串

整型变量 Tlen：文本串长度

整型变量 Plen：模式串长度

整型变量 sumimp_BF：用于统计 I_BF 算法所匹配的总字符数

整型变量 tangimp_BF：用于统计 I_BF 算法所匹配的趟数，初值为 1

字符型变量 firstchar：Pattern 中的首字符，初值为 Pattern［0］

I_BF 算法的思想是：扫描整个文本串 text，查找文本串中与模式串 pattern 的首字符相同位置进行匹配，匹配方式是左匹配，不相同的位置不进行匹配，直到扫描完成。

2.3 I_BF 算法的匹配阶段

在匹配阶段，需要定义如下几个局部变量：

整型变量 k：用来记录目标串 text 的下标值，初值为 0，终值为 Tlen－1。

整型变量 i：用来记录文本串中出现 firstchar 的位置，如果与模式串 pattern 的 j 位置字符匹配，则自增。

整型变量 j：用来记录文本串 pattern 的下标值，初值为 0，如果与字符串 text 的 i 位置字符匹配，则自增。

I_BF 算法的匹配过程是通过判断文本串 text 的当前下标的值是否与模式串 pattern 的首字符相同，若相同则进行匹配，若不相同则不进行匹配。算法的匹配过程描述为：

1）初始化相关变量：$k=0$；

2）当 $k<$Tlen 时执行 3）；否则执行 9）；

3）若 text［k］等于 firstchar，则执行 4），否则执行 8）；

4）进行变量 i 的赋值，即 $i=k$；

5）进行变量 j 的赋值，即 $j=0$；

6）当 i，j 均小于 text、pattern 的长度时，执行 6）；

7）text 的子串 text ′开始与 pattern 进行左匹配，若匹配成功，变量 i、j 均自增，执行 6）；若不匹配，跳出匹配窗口执行 8）；

8）k＋＋，回到 2）；

9）匹配结束。

算法的程序段为：

```
k=0;
while (k<TLen)
  {
if (text[k] == firstchar)
  {
  int i = k;
  int j = 0;
  while(i < Tlen && j < Plen)
  {
  if (text[i] == pattern[j]){
  i++;
  j++;
  }
  else
  break;
  }
  }
k++;
  }
```

为了测试 I_BF 算法，在程序段中添加两个语句，一个是统计 I_BF 算法所匹配的总字符数变量 sumimp_BF 自增功能语句，另一个是用于统计 I_BF 算法所匹配的趟数 tangimp_BF 自增功能语句。修改后的程序段为：

```
k=0;
while (k < TLen)
  {
  if (text[k] == firstchar)
  {
  tangimp_BF++;
  int i = k;
  int j = 0;
  while(i < Tlen && j < Plen)
  {
  sumimp_BF++;
  if (text[i] == pattern[j])
{
  i++;
  j++;
  }
  else  break;
  }
  }
k++;
  }
```

2.4 两种算法的实例比较

为了说明 I_BF 算法性能的优越性，在此通过几个简单实例从字符比较次数及比较趟数两方面对 BF 算法及 I_BF 算法进行比较。

取模式串 P 为“taobao”，长度为 6。

表 1 是对两种算法在比较字符个数方面的实验结果。

表 1 BF 算法与 I_BF 算法比较字符个数

实例	文本串 T	文本串长度	BF（个）	I_BF（个）	I_BF 与 BF 结果比较
1	adffgtwsfslfkatsfxbtaobao	25	27	10	优于
2	aoaoaoaoaoaottttttttttttttttaobaower	35	49	34	优于
3	aotaoaoaoaoaoaobao	18	16	4	优于
4	taoaaaaaabbbbbbbaobaobao	24	22	4	优于
5	oooooooooooooo	15	10	0	优于
6	ttttttttttttttttttttttttttt	27	44	44	相当

表 2 是对两种算法在比较字符趟数方面的实验结果。

表 2　BF 算法与 I_BF 算法比较趟数

实例	文本串 T	文本串长度	BF（趟）	I_BF（趟）	I_BF 与 BF 结果比较
1	adffgtwsfslfkatsfxbtaobao	25	20	3	优于
2	aoaoaoaoaoaoattttttttttttttttaobaower	35	30	15	优于
3	aotaoaoaoaoaoaobao	18	13	1	优于
4	taoaaaaaabbbbbbbaobaobao	24	19	1	优于
5	oooooooooooooooo	15	10	0	优于
6	ttttttttttttttttttttttttttt	27	22	22	相当

表 1、表 2 中，实例 1 至 4 都是匹配成功的实例，当文本串中出现模式串首字符的次数不等于文本串长度时，I_BF 算法总是优于 BF 算法；实例 5 和 6 是匹配失败的实例，其中实例 5 是文本串中不出现模式串首字符，此时 I_BF 算法只需要进行预处理，所花时间也是预处理阶段，而不需要进行匹配，所以此种情况是最优，而实例 6 是文本串中的字符全部都是模式串首字符，此时 I_BF 算法与 BF 算法相当。

在实际应用中，文本串中的字符都是模式串首字符的情况很少，所以，在实际应用中，I_BF 算法的应用性还是较大的。

2.5　I_BF 算法性能分析

I_BF 算法在匹配阶段的跳跃次数完全取决于 firstchar 在 T 中出现的次数，算法最好情况是 T 中没有出现 firstchar，此时时间复杂度是 $O(1)$；最坏情况 T 中所有字符都是 firstchar，即出现 n 次，则此阶段的时间复杂度是 $O(n*m)$。

综上分析，I_BF 算法性能比 BF 算法优。

3　I_BF 算法性能测试

为了检测 I_BF 算法的性能，从不同角度对 BF 算法、I_BF 算法进行测试。测试的操作系统用 Win 7，实现算法的软件是 Visual Studio 2010。

测试文本选用两方面文本进行：

测试一：

文本是随机输入连续的字符串，字符长度为 256960，测试的模式串分两种情况从匹配字符个数、匹配次数、所花时间三方面进行比较，分别是：

第一种情况是模式串在文本串中，也就是匹配成功情况，运行结果如图 1 所示。

图 1　随机字符串匹配成功

图 1 中，右边栏的结果是匹配结果，每两行为一组，每个测试时间为运行 10 万次所取的平均时间。（1）第一组测试结果是模式串为" abc"，长度为 3；（2）第二组测试结果是模式串为" abcde"，长度为 5；（3）第三组测试结果是模式串为" abcdefabcde"，长度为 11。从运行结果可知，I_BF 算法的比较趟数、比较字符个数和匹配时间都比 BF 算法少。

第二种情况是模式串不在文本串中，也就匹配失败情况，如图 2 所示。

图 2　随机字符串匹配失败

三组测试模式串分别是：" zyz"，" xyzwx "" xyzwxvwxsyuu "，长度分别是：3，5，12。由于模式串不在文本串中，所以 I_BF 算法不需要进行比较字符串，所花的时间是在查找首字符，结果比匹配成功时相对于 BF 算法所花时间更少。

测试二：

文本是一篇英文小说，字符长度为 480865，测试的模式串是单词，分两种情况，一种是模式串在文本串中，也就是匹配成功情况。为了更直观地说明 I_BF 算法的优越性，现用表格形式进行两种算法在比较趟数、比较字符个数、运行时间三方面展示，分别如表 3、表 4、表 5 所示；另一种是模式串不在文本串中，也就匹配失败情况，两种算法在三方面的对比综合如表 6 所示。

表 3　匹配成功时 BF 算法与 I_BF 算法比较趟数结果对比

算法	比较趟数		
	模式串长度 3	模式串长度 7	模式串长度 12
BF	480863	480859	480854
I_BF	20802	1878	27327
I_BF/BF	4.33%	0.39%	5.68%

表 4　匹配成功时 BF 算法与 I_BF 算法比较字符个数结果对比

算法	比较字符个数		
	模式串长度为 3	模式串长度为 7	模式串长度为 12
BF	504521	483699	508589
I_BF	44460	4718	55062
I_BF/BF	8.81%	0.98%	10.83%

表 5　匹配成功时 BF 算法与 I_BF 算法运行时间结果对比

算法	运行时间(ms)		
	模式串长度为 3	模式串长度为 7	模式串长度为 12
BF	1.977958	1.185070	1.595256
I_BF	0.775872	0.465954	0.814588
I_BF/BF	39.23%	39.32%	51.06%

从表 3、4、5 可以看出：I_BF 算法在比较趟数、比较字符个数、运行时间，均比 BF 算法有很明显的优势，特别是运行时间方面，约为 BF 算法的 43.2%。在 I_BF 算法中，模式串在文本串中出现的次数越少，所比较趟数和运行时间越少。

表 6 匹配失败时 BF 算法与 I_BF 算法情况对比

匹配类别 / 模式串长度 / 算法	比较趟数			比较字符个数			运行时间(ms)		
	3	14	25	3	14	25	3	14	25
BF	480862	480852	480841	525691	497841	487280	2.057735	1.774223	1.689217
I_BF	30665	15757	5354	75494	32746	11793	0.765518	0.488552	0.385038
I_BF/BF	6.38%	3.28%	1.11%	14.36%	6.58%	2.42%	37.20%	27.54%	22.79%

从表 6 可以看出，当匹配失败情况下，无论模式串长度如何，I_BF 算法所花时间远远少于 BF 算法，约为 25.17%，而且实验发现，模式串长度越大，I_BF 算法运行时间越少。

从以上实验可证明，I_BF 算法是一个性能很好的改进算法，这是因为 I_BF 算法在匹配时只需要考虑 P 中首字符在 T 中出现的位置，这样大大增大了跳跃距离，减少了匹配次数。因此，I_BF 算法效率更高。

4 结论

典型 BF 算法从 T 中的第一个字符开始进行比较，每次匹配都是把 P 的首字符与当前匹配窗口的 T 的下一个字符开始匹配，完全没有利用已取得的部分匹配字符，导致效率低。本文提出的 I_BF 算法的移动距离是根据 P 中首字符在 T 中的位置来确定，匹配方式与 BF 算法一样从左到右。从实验结果和综合分析可知，由于 I_BF 算法能够很大幅度地跳过坏字符，大大减少了移动的次数和比较的字符个数，减少了匹配时间，而且 I_BF 算法在编程方面容易实现，所以 I_BF 算法在实际应用中更实用、更有效。

参考文献：

[1] 蔡 恒，张 帅. 基于 BF 算法改进的字符串模式匹配算法 [J]. 电脑编程技巧与维护，2014 (22)：14－15，33.

[2] 王文霞. BF 模式匹配算法的探讨与改进 [J]. 运城学院学报，2016，34 (6)：63－65.

[3] 朱宁洪. 字符串匹配算法 Sunday 的改进 [J]. 西安科技大学学报，2016，36 (1)：111－115.

[4] 曹海锋，张维琪. 对 Horspool 算法的改进 [J]. 企业技术开发，2015，34 (6)：46－47，69.

[5] 杨 品，吴宇佳，刘嘉勇. 基于 KR－BM 算法的多模式匹配算法改进 [J]. 信息安全与通信保密，2014 (11)：117－120.

[6] 许家铭，李晓东，金 键，等. 一种高效的多模式字符串匹配算法 [J]. 计算机工程，2014，40 (3)：315－320.

[7] 胡桂淼. 多模匹配算法及在入侵检测系统中的应用 [D]. 杭州：浙江工业大学，2014.

[8] 巫喜红. 入侵检测系统中 Wu_Manber 多模式匹配算法的研究 [J]. 计算机应用与软件，2008，25 (8)：114－125.

[9] 陈洪涛. 入侵检测中多模式匹配算法的应用研究 [D]. 天津：天津理工大学，2015.

[10] 李春葆. 数据结构教程 [M]. 北京：清华大学出版社，2013.

(上接第 172 页)

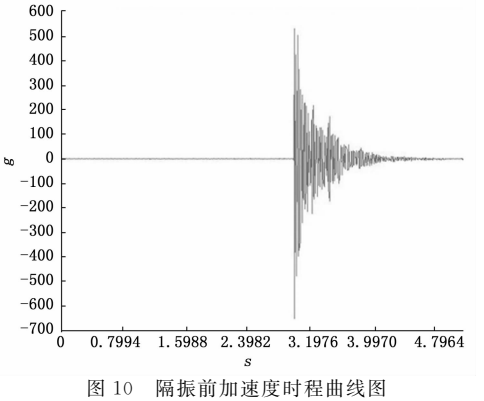

图 10 隔振前加速度时程曲线图

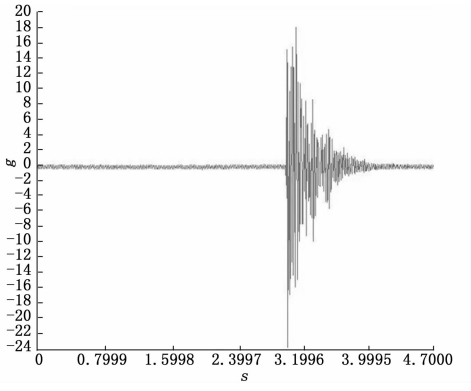

图 11 隔振后垂向加速度时程曲线图

爆炸试验结果为双体船船体结构未出现明显变形，平台垂向加速度峰值衰减率为 95%，爆炸试验结果和数值仿真计算结果吻合良好，验证有限元模型建立准确，平台结构和隔振系统设计合理。

5 结束语

通过水下爆炸冲击平台在水下爆炸载荷作用下的冲击响应数值仿真计算和水下爆炸试验验证，得出以下结论：

1）采用数值仿真计算和水下爆炸试验验证相结合设计水下爆炸冲击平台的方法是合理可行的，水下爆炸载荷作用下平台冲击响应的数值计算结果与试验值吻合良好；

2）水下爆炸冲击平台结构和隔振系统优化设计方法正确，平台抗冲隔振效果满足设计研制要求。

参考文献：

[1] 姚熊亮，张阿漫，许维军，等. 基于 ABAQUS 软件的舰船水下爆炸研究 [J]. 哈尔滨工程大学学报，2006，27 (1)：37－41.

[2] 张阿漫，郭 君，孙龙泉. 舰船结构毁伤与生命力基础 [M]. 北京：国防工业出版社，2012.

[3] 李 琛，张姝红，周学滨，等. 水下非接触爆炸抗冲击缓冲平台性能研究 [J]. 爆破，2011，28 (3)：86－89.

[4] GJB1060. 1－91 舰船环境条件要求机械环境 [S]. 北京：国防科学技术工业委员会，1991.

[5] 科布伦茨. BV0430－85 德国国防军舰建造规范－冲击安全性 [S]. 联邦德国国防装备技术和采购局，1987.

[6] 姚熊亮，徐小刚，张凤香. 流场网格划分对水下爆炸结构响应的影响 [J]. 哈尔滨工程大学学报，2003，6 (3)：238－244.