第 26 卷第 2 期　　四川理工学院学报(自然科学版)　　Vol. 26　No. 2
2013 年 4 月　　Journal of Sichuan University of Science & Engineering(Natural Science Edition)　　Apr. 2013
文章编号:1673-1549(2013)02-0078-04　　DOI:10.3969/j.issn.1673-1549.2013.02.019

一种改进的 Sunday 模式匹配算法

李映刚

(成都理工大学管理科学学院, 成都 610059)

摘　要:字符匹配效率是很多计算机应用系统的性能瓶颈,研究设计高效的匹配算法有助于提高相应系统的应用性能。在分析典型 Sunday 匹配算法的基础上,对其进行了较为有效的改进。改进算法在字符串匹配前先计算模式串的倒序特征值,也就是以此计算出模式串的最后 s 个字符在本模式串中倒序除自己以外的下一次出现的位置。每一次字符匹配都采用倒序匹配并利用这种匹配的结果,匹配结果结合倒序特征值可以直接决定特征串的下一次位移数。在进行完一次字符匹配后,采用增加一个遍历字符的 Sunday 算法来遍历模式串以计算下一次位移数,以此尽可能地排除无效匹配。实验结果表明改进算法的效率比 Sunday 算法有一定提高。

关键词:字符串;模式匹配;倒序字符匹配;Sunday 算法
中图分类号:TP301.6　　**文献标志码**:A

引　言

随着网络应用范围的不断扩大,对实时网络信息的快速搜索的要求也与日俱增。字符串匹配在网络领域有着广泛的应用,比如,拼写检查、语言翻译、数据压缩、搜索引擎、网络入侵检测、DNA 序列匹配等。如何用高效的字符匹配算法解决问题成为目前研究的重要领域之一。

字符串匹配问题的形式化定义是[1-2]:在字符集@上,给定一个长度为 N 的主字符串 $T[1\cdots N]$,以及一个长度为 M 的模式串 $P[1\cdots M]$, $M<<N$。如果对于位置 $S(1<=S<=N-M+1)$,字符串 $T[S\cdots S+M-1]$ 与 $P[1\cdots M]$ 相同,则认定模式串 P 出现在主串 T 的 S 位置。字符串匹配问题就是要寻找 P 在 T 中是否出现以及出现的位置。

1　相关算法简介

1.1　BF 算法

BF(Brute Force)算法是最原始且效率最低的算法。它从主串第一个字符开始逐个与模式串的每个字符相比较是否相同,如果不匹配,然后从主串下一个字符重新与模式串作相同性比较。BF 算法原理简单、易实现,但效率太低。

1.2　KMP 算法

KMP(Knuth – Morris – Pratt)[3-4] 算法是 D. E. Kunth、J. H. Morris 和 V. R. Pratt 等人在 1977 年提出的。KMP 算法在一次匹配失败后可以利用部分匹配的结果直接将模式串向右移动尽量远的距离来进行下一次匹配。KMP 算法虽然对 BF 算法有了很大的改进,但是原理复杂,不易推广实现。

1.3　BM 算法

BM 算法[4-5]是 Boyer 和 Moore 提出的一种算法。最主要的特点就是在匹配过程中,可以跳过很多无用的字符,这种跳跃式的匹配可以获得很高的匹配效率。

1.4　ZZL 算法

ZZL 算法[6]是纪福建等人提出的一种可用作特殊用途的字符串匹配算法。ZZL 算法的核心思想是:

收稿日期:2013-03-07
基金项目:中国地质调查局地质调查工作项目(1212011085536)
作者简介:李映刚(1986-),男,四川绵阳人,硕士生,主要从事嵌入式 Linux 系统方面的研究,(E-mail)bestwill@126.com

首先在主串 T 中查找模式串 P 的首字母,每找到一个则将它的位置存储,然后依次提取这些位置,从这些位置来匹配模式串 P。对于频繁使用的主串和模式串,匹配速度会非常快,缺点是可能会需要大量的存储空间来存储那些频繁使用的模式串的首字母的位置。

1.5 Sunday 算法

Sunday 算法[5,7]在匹配过程中并不要求字符比较的方向,在一次匹配完后,利用主串的下一个字符逆序搜索模式串,找到这个字符第一次出现的位置或者不出现。从而达到利用一个字符的遍历就能决定模式串的后移位置,极大的提高了移动效率。其核心思想就是用较少的字符比较来确定较长的模式串的位移。

一般情况下,BF 算法效率最低,KMP 算法与 BM 算法原理复杂不易实现,Sunday 算法原理简单执行速度最快。本文就选取快速的 Sunday 算法进行改进。Sunday 算法有两个缺点:首先就是对前一次匹配结果没有进行有效的利用;其次就是 Sunday 算法利用主串下一个字符来遍历模式串的作法也过于简单。当模式串很长的时候(很多实际应用中,模式串都比较长),主串下一个字符在模式串中出现的概率会提高,这样的出现会增加很多的无效匹配。针对 Sunday 算法的两个缺陷,提出了将原理简单易实现的前一次比较结果再利用方法与主串下一个字符搜索加强方法相结合的思想,以此来改进 Sunday 算法,以期得到更高的匹配效率。

2 改进 Sunday 算法思路

2.1 前一次比较结果简单再利用

得益于 BM 算法的逆序比较思想,提出一种简单易理解的逆序比较结果再利用的算法。

算法描述:将模式字符串 $P[1\cdots M]$ 与主字符串 $T[1\cdots N]$ 在一次匹配比较过程中,按照字符串的逆向进行比较。从模式字符串的最后一个字符与主串相应位置开始,依次向前比较并最后确定匹配字符的数目(记为 A)。根据模式串的特性,最后 A 个字符组成的模式串 P 的子串(记为 $SP[A]$),从模式串后面往前遍历,可以找到的第一个再次出现的子串 $SP[A]$(本身除外)的位置(记为 S),这一过程可以在匹配前作预处理,作为模式串的特征值记录下来。如果主串与模式串从后向前最多 A 个字符相同,那么根据预处理的特征值的位置 S,就可以不用进行字符比较,而直接决定下一次模式串的移动距离。

例如,模式串为“facdefa”,主串为“ekjacfadbcea……”,进行第一次匹配比较,倒序比较发现模式串的最后两个字符“fa”和主串的相应位置的字符相同,而“fa”在模式串中倒序再次出现的位置为 1,那么可以直接决定模式串在主串中向后移动 5 步的距离,而不用多余的比较。算法优点就是仅仅利用预处理的结果就能决定移动步数,是一种原理简单易实现的前一次匹配结果再利用方法。算法缺点就是预处理数据能否有效果要看模式串的最后一个字符在主串中出现的概率,就平均情况而言,这与字符集@ 的大小有关。

2.2 增加 Sunday 算法中的遍历模式串的字符数

Sunday 算法的核心体现在 P 与 T 进行比较确定是否匹配后,用主串下一个字符来遍历模式串进行比较就能确定模式串下一次右移的距离,仅仅通过增加遍历模式串的字符数目,就能极大的提高移动效率。比如增加一个遍历字符,在进行一次匹配后,用主串后两个字符来遍历模式串,进行比较来确定模式串下一次右移的距离。能极大提高移动步数的原因就是两个字符的比较不仅仅包含是否相等,还包含了他们之间的位置关系,位置关系所包含的信息是不用浪费时间去比较的,但是却能用来辨别模式串的移动距离。模式串越长 Sunday 算法的下一个字符在模式串中出现的概率就越大,无效匹配的次数就越多,由此可以预见,对于此改进方法,模式串越长,效率越高。

2.3 改进算法的处理过程

(1) 模式串预处理

在进行字符匹配前,先建立一个数组 $fea[M-1]$,M 为模式串字符数,用于记录模式串的倒序特征值。

例如,模式串为 P =“afkdfk”。那么倒序 s 个字符在模式串中除自己以外的下一次出现位置就是:

$s=1, p[0]=3$;

$s=2, p[1]=2$;

$s=3, p[2]=0$;

$s=4, p[3]=0$;

$s=5, p[4]=0$;

那么 $fea[5]=\{3,2,0,0,0\}$。如果一次倒序匹配中,匹配到的最多字符数目为 s,那么直接决定模式串的下一次位移数就为 $D1=M-s+1-fea[s]$。

(2) 字符倒序匹配

记录下倒序匹配得到的最大字符数 s,利用位移公式得到下一次位移数 $D1$。

(3) 增加 Sunday 算法的遍历模式串字符数

利用一次字符匹配后的主串的后 x 个字符来遍历模式串,思想与 Sunday 算法是一致的,只有在这 x 个字符与模式串中的某 x 个字符完全一致时,才能确定模式串的下一次位移数,记为 $D2$。

(4) 选取 D1 与 $D2$ 中较大的一个作为模式串的后移步数 D。

(5) 进行下一次字符倒序匹配,重复(2)。

3 改进算法的性能分析

3.1 性能分析

不考虑对模式串的预处理过程,统计算法运行过程中总的字符比较次数,并作为算法的性能指标。倒序比较结果再利用算法的效率与模式串的最后一个字符在主串中出现的概率有关,出现概率越大,效率越高;通过将遍历模式串的字符数增加到 x 个,那么在模式串中出现这 x 个字符并且顺序一致的情况将大为减少,能提高匹配效率。同时,当模式串越长时,两种思想发生效果时后移的步数就越大,效果越好。

3.2 实验与结果

为了测试改进算法的性能,先用简单的数据来分析验证改进算法的效率。假如得到模式串为 $P=\{a,k,f,a\}$;主串为 $T=\{a,j,f,l,d,s,j,f,s,f,a,d,a,k,f,a\cdots..\}$。那么当模式串在与主串第 7 个子串进行匹配后,可以直接得到下一次模式串位移数 $D1=3$;当模式串与主串第 9 个子串进行匹配后,可以直接得到下一次模式串位移数 $D1=3$。模式串与主串第 4 个子串匹配后,若采用常规的 Sunday 算法,可以得到的下一次模式串位移数 $D2=2$,但是若采用遍历字符数 $x=2$ 的改进算法,可以得到的下一次模式串位移数 $D2=5$。可见改进算法效率是有提高的。

现在用电脑随机生成长度为 N 的主串,以及长度为 M 的模式串,匹配到模式串的数目记为 S,遍历模式串的字符数 $x=3$。运行并统计相关算法的总的字符匹配数目。结果统计见表 1。

表 1 五种算法的匹配测试结果

$N=8192$	BF	Sunday	改进思路 1		改进思路 2		改进 sunday 算法	
			次数	百分比	次数	百分比	次数	百分比
$M=4$ $S=154$	32756	14106	14106	1.00	14119	1.00	14119	1.00
$M=8$ $S=138$	65480	15972	15970	0.99	15709	0.98	15705	0.98
$M=16$ $S=0$	130832	17751	17722	0.99	17286	0.97	17258	0.97
$M=32$ $S=0$	261152	23713	20708	0.87	18102	0.76	17986	0.75
$M=64$ $S=5$	520256	30476	25128	0.82	23360	0.76	22611	0.74
$M=72$ $S=0$	584712	43263	21509	0.49	20222	0.46	18320	0.42
$M=96$ $S=0$	777312	51692	25415	0.49	28644	0.55	25092	0.48
$M=116$ $S=0$	936932	47285	27426	0.58	31635	0.66	26298	0.55
$M=130$ $S=0$	1048190	72678	20334	0.27	56855	0.78	20246	0.27

注:百分比是本次改进相对于 Sunday 算法的比较次数的比率

对表 1 中的数据分析,说明两种改进思路得到的算法在模式串较短的时候与 Sunday 算法性能持平,而随着模式串的增长,效率显著提升。基于这两种思路的改进算法,在模式串长度较大时,效率明显优于 Sunday 算法。

4 结束语

改进的 Sunday 算法在原 Sunday 算法的基础上增加了一个模式串的倒序特征值,利用这个预先得到的特征值可以直接决定下一次的移动步数;同时也增加了主串的遍历模式串时的字符个数,减少了很多无效匹配。实验结果表明,改进的 Sunday 算法提高了字符匹配的效率。

参 考 文 献:

[1] 曾慧惠,袁世忠,胡 鹏.入侵检测系统中高效模式匹配算法的研究[J].计算机应用于软件,2008,25(4):226-227.

[2] 徐 珊,袁小坊,王 东,等.Sunday 字符串匹配算法的效率改进[J].计算机工程与应用,2011,47(29):96-98,160.

[3] 周延森,汪永好.网络入侵检测系统模式匹配算法研究[J].计算机工程与设计,2008,29(7):1652-1654,1683.

[4] 万晓榆,杨 波,樊自甫.改进的 Sunday 模式匹配算法[J].计算机工程,2009,35(7):125-126,129.

[5] 王 成,刘金刚.一种改进的字符串匹配算法[J].计算

机工程,2006,32(2):62-64.

[6] 赵俊杰.一种用于关键词检索的快速字符串精确匹配算法[J].计算机系统应用,2010,19(2):189-191.

[7] 潘冠桦,张兴忠.Sunday 算法效率分析[J].计算机应用,2012,32(11):3082-3084,3088.

An Improved Sunday Pattern Matching Algorithm

LI Ying-gang

(School of Management Science, Chengdu University of Technology, Chengdu 610059, China)

Abstract: The efficiency of character matches are the performance bottlenecks of many computer applications, an algorithm which improves the efficiency of the characters match can contribute to enhancing the application performance of the system. Based on the analysis of Sunday pattern mathching algorithm, many improvements have been done. This improved algorithm calculation the mode string's Reverse Eigenvalue before string matching so that it can calculate the reverse position beside themselves for characters of the last *s* mode string. Every character matching is the descending match and got by the result of it, in which combination the result of the descending match and the Reverse Eigenvalue can directly determine the next displacement number of the mode string. After once character matching, the mode string is traversed to calculate the next displacement number by Sunday algorithm(it uses one character). which can exclude many invalid match by simply increasing the number of traversal characters. Experimental results show that the efficiency improved algorithms to Sunday algorithm has definitely improved.

Key words: string; pattern matching; descending character matching; Sunday algorithm