打字比赛系统中字符串匹配算法①

李载源

(长安大学 电子与控制工程学院, 西安 710064)

摘 要: 打字比赛程序, 是学校为了加强学生对计算机基本技能的掌握, 而开发的一款用于检验打字速度、准确率等多项指标的比赛程序. 传统的字符串匹配算法是基于关键词的匹配算法, 只能得到位置信息, 而比赛程序是两个大文本之间进行匹配, 需要获得更加详细的匹配信息, 这时传统算法就无法满足要求. 通过对实际问题的探索和研究, 在传统算法的基础上, 设计出一种新的匹配算法. 这种算法可以进行两个文本之间的匹配, 得到正确的内容、错误的内容等信息. 应用这一算法编写软件, 验证了算法的可行性和实用性.

关键词: 打字比赛; 关键词; 大文本; 字符串匹配; 算法

String Matching Algorithm in the Program of Typing Match

LI Zai-Yuan

(Electronic and Control Engineering of Chang'an University, Xi'an 710064, China)

Abstract: The program of typing match is a game program which used to test many indicators of typing, like speed, accuracy and so on. Many schools use it to strengthen students mastery of basic computer skills. The traditional algorithm is based on the keywords matching algorithm, only can get the location information, but the game program is the program of matching between two large text segment, it need get more matching information, so the traditional algorithm is unable to meet the requirement. On the basis of traditional algorithm, a new matching algorithm was designed by the exploration and research of the practical problems. This algorithm can match between the two text, and get the correct content, the error content and other information. So wrote a software in this algorithm, proved the feasibility and practicality of this algorithm.

Key words: typing match; key words; large text segment; string matching; algorithm

1 引言

字符串匹配问题在打字程序、文字编辑、图像处理、语言识别、文献检索、网络安全等领域有着广泛的应用, 因此探索研究一种能够快速匹配字符串并能满足实际需要的方法意义重大. 传统的字符穿匹配算法是基于关键词的匹配算法, 所谓关键词匹配算法[1], 即给定一个文本内容和模式串, 在给定的文本内容中运用各种算法, 找出所有与模式串精确匹配的子串起始位置、结束位置、长度等信息的一类算法. 这类匹配算法能够满足字符串匹配时对速度的要求, 在一些实际问题的处理上却无满足实际需要. 比如, 打字比赛系统中要求输入的一个文本段与原文本段进行比较, 找到两者之间的联系信息(输入正确的信息、错误的信息、多输入信息、少输入信息……), 显然传统算法达不到上述要求.

本文对部分传统算法进行了详细分析介绍, 找出了传统算法的优点和不足, 在此基础上根据实际需求提出了一种改进算法, 实现了大文本之间的匹配, 匹配结果能获得更多的匹配信息.

2 传统匹配算法

简单的匹配算法[2,3]是基本的匹配算法, 如果匹配不成功模式串移动距离始终为 1, 因此简单的匹配算法匹配次数多, 时间复杂度较大. 经过了几十年发展, 字符串匹配算法有了很大的改进. 它们通常以长度为

① 收稿时间:2013-07-11;收到修改稿时间:2013-09-09

m 的模式 P 为参照, 在文本 T 中用不同的方法进行匹配, 在很大程度上减少了匹配次数, 传统的匹配算法有 KMP, BM, Sunday 等.

2.1 KMP 算法

KMP[4-6]是从左到右的前缀匹配算法, KMP 算法的核心思想是通过之前匹配得到的部分信息来进行之后的匹配过程. 用 KMP 算法需要一个模式值(*next*[*n*]), *n* 表示模式串字符的下标, 假定 *S* 为原文本, *T* 为模式串, 模式值的获取暂且定义如下:

(1) *next*[0]= -1 模式串第一个字符的模式值为-1.

(2) *next*[*j*]= -1 *T*[*j*]==*T*[0]且 *T*[0]*T*[1]⋯*T*[*k*-1]≠*T*[*j*-*k*]*T*[*j*-*k*+1]⋯*T*[*j*-1](或相等但 *T*[*k*]=*T*[*j*])(1≤k<j).

(3) *next*[*j*]=k *T*[0]*T*[1]⋯*T*[*k*-1]==*T*[*j*-*k*]*T*[*j*-*k*+1]⋯*T*[*j*-1]且 *T*[*j*]≠*T*[*k*](1≤k<j).

(4) *next*[*j*]=0 除(1)(2)(3)以外的其他情况.

若模式串 T="abcac"则它的模式值如表 1 所示.

表 1 模式串对应的模式值

下标	0	1	2	3	4
T	a	b	c	a	c
next	-1	0	0	-1	1

根据模式值进行匹配的过程如表 2 所示. 其中黑体表示进行过匹配操作的字符.

表 2 KMP 算法匹配过程

	0	1	2	3	4	5	6	7	8	9	10	11	12
文本串	*e*	*a*	*b*	*a*	*b*	*q*	*a*	*b*	*c*	*a*	*c*	*b*	*a*
第一次	*a*	*b*	*c*	*a*	*c*								
第二次		*a*	*b*	*c*	*a*	*c*							
第三次				*a*	*b*	*c*	*a*	*c*					
第四次						*a*	*b*	*c*	*a*	*c*			
第五次							*a*	*b*	*c*	*a*	*c*		

2.2 BM 匹配算法

BM[7]算法是通过对 KMP 算法的改进而得到的一种算法. 与 KMP 算法相比匹配操作的方向不同, BM 算法改为从右向左, 即从模式串的末字符开始匹配, 开始匹配时模式串 *P* 的最左端与原文本 *S* 最左端对齐, 当某一趟匹配中出现匹配失败时, 根据匹配信息计算模式串需要进行右移的距离, 右移之后再从右到左继续匹配, 其中右移的距离有专门的移动距离控制函数进行控制. 算法增加了窗口的移动距离, 跳过了很多不需要的字符. 为了更好的介绍 BM 算法现作如下定义.

模式串 T="t_1 t_2 t_3 ⋯t_m"

C={*c*|*c* 在模式串 *T* 中出现} *c* 是文本串中字符

dist: *c*→{1,2,⋯, *m*}

dist(*c*) = *m*-*j* j 为 c 在模式中的下标

dist(*c*) = *m*+1 若 *c* 不在模式中或 *c* = t_m

其中移动距离控制函数(dist)为滑动距离函数, 它是一个从字符到正整数的映射. 具体的匹配过程如表 3 所示. 黑体表示进行过匹配操作的字符.

表 3 BM 算法匹配过程

	0	1	2	3	4	5	6	7	8	9	10	11	12
文本串	*e*	*a*	*b*	*a*	*b*	*q*	*a*	*b*	*c*	*a*	*c*	*b*	*a*
第一次	*a*	*b*	*c*	*a*	***c***								
第二次				*a*	*b*	*c*	*a*	***c***					
第三次							*a*	*b*	*c*	*a*	*c*		

2.3 Sunday 算法

Sunday[8-10]算法跟 BM 算法很相似, 它不要求从左向右匹配还是从右向左匹配, 不管从哪个方向匹配当它发现匹配失败时, 算法下一步关注的是文本串中参与匹配的最后一个字符串的下一个字符, 如果判断该字符在匹配串中没有出现, 则移动距离=被匹配的文本串长度+1, 否则, 算法跟 BM 算法一致. Sunday 算法的匹配过程如表 4 所示.

表 4 Sunday 算法匹配过程

	0	1	2	3	4	5	6	7	8	9	10	11	12
文本串	*e*	*a*	*b*	*a*	*b*	*q*	*a*	*b*	*c*	*a*	*c*	*b*	*a*
第一次	*a*	*b*	*c*	*a*	*c*								
第五次							*a*	*b*	*c*	*a*	*c*		

2.4 对传统算法的总结

由 2.1-2.3 的介绍, 可以看出, 传统算法都是基于关键词的匹配算法, 在满足关键词匹配的基础上尽可能减少匹配次数, 以达到快速匹配的目的. 算法匹配次数如表 5 所示.

表 5 各算法匹配次数

匹配算法	简单匹配算法	KMP	BM	Sunday
匹配次数	17	14	7	6

传统匹配算法有很多共同的特点:

(1) 需要确定一模式串 *T*;

(2) 确定的模式串 *T* 的长度小于文本串 *S*, 否则匹配没有意义;

(3) *T* 中的字符需要在 *S* 中完全匹配.

(4) 通过辅助的标记函数减少匹配次数, 达到最

优匹配.

由以上传统算法的特点可以看出, 传统算法在匹配速度上优势突出, 但他们也有一定的局限性, 因为前三个特点决定了传统算法无法进行模糊匹配[11,12](模式串 T 在文本 S 中的不完全匹配), 这种情况多出现在模糊查询中. 在本文要讲述的打字比赛程序中, 文本串是确定的, 模式串的大小内容却是变化的, 不确定的, 如此就无法使用传统的匹配算法. 这类问题下面会详细介绍.

3 需求分析

打字比赛时的要求是原文为 S=[1,2,…,n], 对照输入为 T=[1,2,…,k], 其中 k 与 n 没有大小上的确定关系, 这时要确定 T 与 S 中的匹配信息(正确字符数、错误字符数、漏字符数……), 然后根据这些指标计算比赛成绩. 为了方便讲述, 对原文 S, 输入文本 T 假定如下.

原文 S=“一个 9 个月前买了一台手机的顾客有可能在一个月内买一个新的电脑我”.

输入 T=“我一个 9 个月买了一个手机的骨科可能在一个月内买了个新电脑新的电脑”

要解决这类实际问题, 如果使用传统匹配算法, 首先遇到的就是模式串如何选择. 如果从文本 S 中选择模式串, 模式串选择太小, 不仅计算量会很大, 而且会匹配出错, 如果模式串选择太大, 则会漏掉可以匹配的部分信息. 比如选择模式串 P=“一”则 T 中有多个匹配成功, 无法确定那个才是最合理的匹配, 匹配不合理对后续匹配影响很大. 如果 P=“一台手机”则找不到匹配项, 然而这种情况其实是因为输入文本 T 中“一个手机”中的“个”字打错了造成的, 这种情况只能说明错了一个字, 不能认定全部错误. 如果从 T 中选择模式串, P=“一”或者 P=“一个手机”, 则会出现同样的问题.

当文本段与文本段之间的匹配不能简单机械的进行匹配, 还要考虑语法先后, 显然传统算法无法满足实际需求, 因此要对算法作进一步的探索和研究.

4 算法设计

算法的软件实现是用 C#语言编写的, 要用到动态数组的存储, 即 Arraylist 和 IEnumerator 数据类型, 本文重点阐述算法在此对 C#语言的相关语法就不做过多介绍.

4.1 算法介绍

首先定义两个 *Arraylist* 类型的数组 *match_all*、*match_right*, 用于动态存储匹配后的内容. 将 T 中的每一个内容与 S 中的全部内容进行比对, 如果比对正确, 将结果信息存储在 *match_all* 中, *match_all*=[T 中匹配项位置, S 中被匹配成功项位置, T 中匹配项内容, S 中被匹配成功项内容]. 然后对 *match_all* 中的内容进行筛选, 在对 *match_all* 中的内容进行操作之前, 先把 *match_all* 中的内容转存到 *IEnumerator* 类型的 *IEnum_all* 中, 利用 *IEnumerator* 中类似指针按照先进先出原则逐一取出其中的内容以便进行操作. 为了方便叙述现做如下定义: a, b, c, d 为比较子串

a, b 存放第一次取出的数据, c, d 存放第二次取出的数据.

a, c=(T 中匹配项位置)

b, d=(S 中被匹配成功项位置)

首先把第一次取出的数据, a、b 及其相应内容存入 *match_right* 中, a、b 则作为上次存储的数据表示. 经过观察发现,

(1) 如果 a=c, 说明 T 中的一项与 S 中的多个匹配成功, 那么 a、b 即为匹配成功最靠前的一组, 说明 c、d 为多余匹配, 存储保留 a、b, 对 c、d 不做操作.

(2) 如果 b=d, 说明有多个 T 中的项与同一个 S 中的项匹配成功, 那么 a、b 为匹配成功最靠前的一组, 说明 c、d 为多余匹配, 对 c、d 不做操作.

(3) 如果 $a\neq c$ 且 $b<c$ 且 $b<d$, 说 a、b 组的匹配成功与 c、d 组的匹配成功没有相互影响, 那么 a、b、c、d 均为正确匹配组, 存储 c、d 及其相应内容, 然后 a=c, b=d, 把本次存储的内容暂存到 a、b 中以供下次比较使用.

(4) 如果 $a\neq c$ 且 $b<c$ 且 $b>d$, 这种情况说明 c、d 的匹配成功对 a、b 造成了影响, 因为 T 中 c 项比 S 中的 b 项靠后, 这里判断 c、d 多余匹配, 存储保留 a、b 的值, 对 c、d 不做操作.

(5) 如果 $a\neq c$ 且 $b>c$ 且 $b<d$, 这种情况说明 a、b 组的匹配成功与 c、d 组的匹配成功相互不影响, 则 a、b, c、d 两组均为正确匹配, 存储 c、d 及其相应内容, 然后 a=c, b=d, 把本次存储的内容暂存到 a、b 中以供下次比较使用.

(6) 如果 $a\neq c$ 且 $b>c$ 且 $b>d$, 这种情况说明匹配成功的两组之间相互有影响. 因为 S 中的 b 比 c、d 组的任何项都靠后, 相对与 c、d 组来说, a、b 组匹配错误, 所以要删除上次存储的 a、b, 存入 c、d 及其相应内容, 然后把本次存储的内容暂存到 a、b 中以供下次比较使用.

(7) 如果 $a\neq c$ 且 $b=c$ 且 $b<d$, 同(5).

(8) 如果 $a\neq c$ 且 $b=c$ 且 $b>d$, 这种情况说明 c、d 的匹配成功对 a、b 造成了影响, 这种情况取舍比较困难, 根据经验观察, 保留 a、b 组比较合理, 对 c、d 不做操作.

如此迭代判断到最后, *match_right* 中存储的就是最终匹配结果. 根据 *match_right* 中存入的内容与 S、T 进行比较, 多输入、少输入、错输入这些信息就不难找出了.

算法流程图如图 1 所示, 图中操作(1-8)的详细内容同上对应.

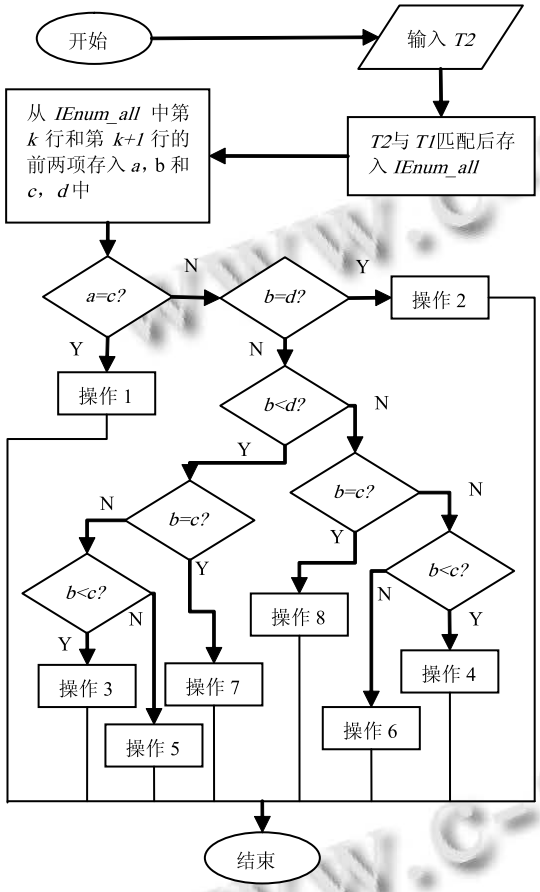

图 1 匹配算法流程图

4.2 与传统算法的比较

根据上面对新算法的详细介绍, 可以得出新算法的一些特点.

不需要选择模式串, 避免了模式串选择带来的局限性. 匹配时只需要确定比较子串 *abcd* 即可.

匹配信息中加入了位置信息, 由于位置信息参与匹配, 匹配具有语法特性, 匹配结果信息更全面.

需要匹配结果信息比较全面, 算法处理速度低于传统匹配算法.

通过对比 2.4 中传统算法的特点, 新算法在处理速度上不如传统算法, 因此新算法在处理速度上还有待进一步的优化设计. 但新算法在应对实际问题时比较灵活. 新算法加入位置信息标记, 面对更多复杂要求时有很大的应对和优化空间. 比如原文“谦虚谨慎, 戒骄戒躁”输入“戒骄戒躁, 谦虚谨慎”这种类型匹配时有两种情况, 1)“戒骄戒躁”正确匹配, 戒骄戒躁之前的“谦虚谨慎”属于少输入的, 之后的“谦虚谨慎”属于多输入的; 2)“谦虚谨慎”正确匹配, 类型同 3. 解决上述两种问题, 通过改变算法(4)和(8)步骤中的取舍就可以实现.

5 设计检验

本设计采用 visual stidio 2005 编程软件, 通过了需求分析、功能模块化、界面设计、软件编写, 使用新算法, 设计编写一款打字比赛程序. 最后测试结果符合需求和预期目标.

例句匹配测试界面如图 2, 上部分是原文, 下部分是输入框. 例句匹配测试结果如图 3 所示. 匹配测试结果综合信息如图 4 所示.

由图 3 匹配测试结果可以看出, 匹配结果正确符合实际要求. 图 4 中显示的是原文字数 31, 正确字数, 24 错误字数 8, 缺少字数 7 . 打错的字分别是“我……手……骨科……了……新电脑”(由于需求原因, 程序中未予以显示). 这里显示的信息只是根据实际需求列出的, 由本文算法(1)至(8)步骤可以看出匹配结果附带了位置信息, 有了这些位置信息, 可以根据需求得到更多的匹配信息.

6 结语

随着高校计算机专业的不断增加, 计算机专业学生越来越多, 为了激发学生对学习计算机学习的热情, 实践证明, 计算机基本技能大赛是个很成功的选择, 计算机基本技能大赛是教育上的大胆创新, 对计算机专业新生而言, 大赛让他们拥有了竞争意识, 紧迫意识, 精益求精意识等, 并为之后的学习打下坚实的基础. 大赛中最主要的是打字比赛, 而打字比赛软件让大赛更加便利、公平、公正.

本文比赛软件, 不仅实现了两汉语文章之间的比对, 也可以对英文文章进行匹配, 英文匹配之前, 需要把英文文章中单词、符号、标点等区分开来、分别存取. 然后使用本文算法进行匹配. 经实践检验本文算法具有一定的灵活性和实用性.

图 2 例句匹配测试界面

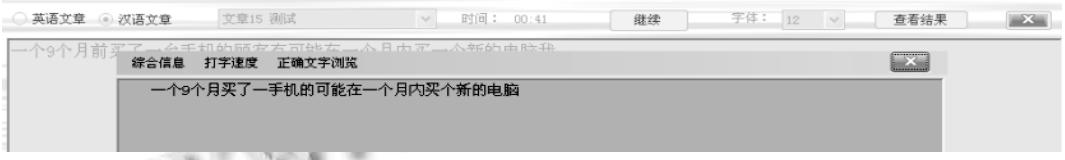

图 3 例句匹配测试结果显示

图 4 例句匹配测试综合信息显示

参考文献

1 Knuth DE, Moms JH, Pratt VR.Fast pattern matching in string. SIAM J Comput, 1977, 6(1): 323–350.

2 Boyer RS, Moore JS. A fast string searching algorithm. Communications of ACM, 1977, 20(10): 762–772.

3 Bai Y, Kobayashi H. New string matching technology for network security. Proc. of the 17th International Conference on Advanced Information Networking and Applications (AINA'03). 2003.

4 Boyer RS, Moore JS. A fast string searching algorithm. Communications of the ACM, 1997, 20(10): 762–772.

5 Aho A, Corasick M. Efficient string matching: An aid to bibliographic search. Communi cations of the ACM, 1975, 18 (6): 333–343.

6 杨薇薇,廖翔.一种改进的 BM 模式匹配算法.计算机应用, 2006,26(2):318–319.

7 王昕阳.浅析串模式匹配算法 KMP 及应用.电脑学习, 2007,(4):40–41.

8 赵念强,鞠时光.入侵检测系统中模式匹配算法的研究.微计算机信息,2005,21(3):22–24.

9 刘燕兵.串匹配算法优化技术研究[学位论文].北京:中国科学院计算技术研究所.2006.

10 王成,刘金刚.一种改进的字符串匹配算法.计算机工程, 2006,32(2):62–64.

11 陈开渠,赵洁,彭志威.快速中文字符串模糊匹配算法.中文信息学报,2004,18(2):58–65.

12 王成,刘金刚.一种改进的字符串匹配算法.计算机工程, 2006,32(2):62–64.