串的最大匹配算法

林毓材 张建军 向永红 张春霞
（云南师范大学计算机科学系，昆明，650092）

摘要：给定两个串S和T，长分别m和n，本文给出了一个找出二串间最大匹配的算法。该算法可用于比较两个串S和T的相似程度，它与串的模式匹配有别。
关键词：模式匹配 串的最大匹配 算法

Algorithm on Maximal Matching of Strings

Lin YuCai Xiang YongHong Zhang ChunXia Zhang JianJun
（Computer Science Department of Yunnan Normal University Kunming 650092）
ABSTRACT Given Two Strings S of length m and T of length n，the paper presents an algorithm which finds the maximal matching of them. The algorithm can be used to compare the similarility of the two strings S and T, it is different with the strings' pattren matching.
KEY WORDS Pattern Matching Maximal Matching of Strings Algorithm

1 问题的提出

字符串的模式匹配主要用于文本处理，例如文本编辑。文本数据的存储（文本压缩）和数据检索系统。所谓字符串的模式匹配[2]，就是给定两个字符串S和T，长度分别为m和n，找出T中出现的一个或多个或所有的S，在这方面已经取得了不少进展[3][4][5][6][7][8][9][10][11]。本文从文本处理的另一个角度出发，找出两个串的最大匹配，比较其相似程度[1]。它主要应用于文本比较，特别是在计算机辅助教学中。显然前者要找S的完全匹配，而后者并无此要求。例如，若S=ABCD，T=EFABCDX，那么模式匹配的结果就是找出了T中的一个ABCD，而我们算法的结果就是S能与T的ABCD完全匹配，但是T中还有3个字符是比S多出来的，也就是说在S中有100%的字符与T中的匹配，而在T中有57%的字符与S中的匹配。若S=ABCDFE，T=AFXBECDY。则在模式匹配中S与T无匹配项，但在我们的算法中就能发现T中存在A，B，C，D，但D后不存在E，F。而且S中也存在A，B，C，D，且具有顺序性。这样就能公正地评价S，T的区别。得知其相似程度。
文章的组织如下：首先介绍基本定义和问题的描述；第三节是算法设计；最后是本文总结。

2 问题的描述

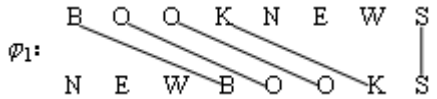
设Σ为任意有限集，其元称为字符，w：Σ→N为Σ到N的函数，称为Σ的权函数（注：本文仅讨论权值恒为1的情况）。Σ^*为Σ上的有限字符串集合，那么对任意S，T∈Σ^*，设$S=a_1a_2\cdots a_m$，$T=b_1b_2\cdots b_n$，m>0，n>0。记<m>={1,2，…,m},<n>={1,2，…,n}，则称{(i,j)∣i∈<m>，j∈<n>，$a_i=b_j$}为S与T的匹配关系集，记作M（S，T），称$\varphi\subseteq M$为S与T的一个（容许）匹配，若对任意（i,j），(i',j')∈φ，① i< i'，当且仅当j< j'，② i= i'当且仅当j= j'。S与T的匹配中满足 $\sum_{(i,j)\in\varphi} w(a_i)$ 最大者，称为S与T的最大匹配。若C(i,j)为N上的m×n矩阵，且满足：

$$C(i,j)=\begin{cases}0, & \text{若 } (i,j)\notin M(S,T);\\ w(a_j), & \text{若 } (i,j)\in M(S,T)。\end{cases}$$

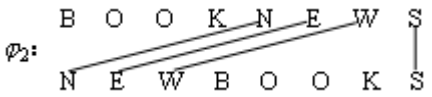
则称矩阵C为串S与T的匹配关系阵。

于是求串S与T的最大匹配φ，等价于求C中的一个最大独立点集M，它满足，若$c_{i,j}$，$c_{i',j'}\in M$，则i< i' 当且仅当j< j'，i=i'当且仅当j=j'。我们称这样的最大独立点集为C的最大C-独立点集。

例：设Σ为所有字母的集合，对任意x∈Σ，w（x）≡1，设S与T分别为：S=“BOOKNEWS”，T=“NEWBOOKS”。则我们可以得到S与T两个匹配：



这里 $\sum_{(i,j)\in\varphi_1} w(a_i)$=5；



这里 $\sum_{(i,j)\in\varphi_2} w(a_i)$ =4。

显然φ_1为串S与T的最大匹配。

S与T的匹配关系阵C可表示如下：

$$C=\begin{array}{c} \begin{matrix} N & E & W & B & O & O & K & S \end{matrix} \\ \begin{pmatrix} 0 & 0 & 0 & ① & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & ① & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & ① & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & ① & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & ① \end{pmatrix} \end{array} \begin{matrix} B \\ O \\ O \\ K \\ N \\ E \\ W \\ S \end{matrix}$$

其中带圈的部分为一最大C-独立点集。

3 算法设计

我们仅就权值为一的情况进行讨论。

设S和T为任意给定串，C为的S与T匹配关系阵，那么由2的讨论知，求S与T的最大匹配问题，等价于求C的最大C-独立点集问题。因而，为了解决我们的问题，只要给出求C的最大C-独立点集的算法就可以了。

显然，为了求出C的最大C-独立点集，我们可以采用这样的方法：搜索C的所有C-独立点集，并找出它的最大者。这种方法是可行的，但并不是非常有效的。这会使问题变得很繁，复杂度很大。因此，我们先对问题进行分析。

在下面的讨论中，我们把C的任一C-独立点集$\varphi=\{a_{i_1,j_1},\cdots,a_{i_s,j_s}\}$，记作$\varphi=a_{i_1,j_1}\cdots a_{i_s,j_s}$，$i_1<\cdots<i_s$。于是$\varphi$可看作阵C中以1为节点的一条路，满足：对路中的任意两节点，均有某一节点位于另一节点的右下方。称这种路为右下行路。

于是求C-独立点集等价于求阵C的右下行路。这种求右下行路的搜索可以逐行往下进行。

命题1. 若φ =á$a_{i,j}$â和ø=á'$a_{i,j}$ó为C的两个C-独立点集，且á为á'的加细，则存在C-独立点集φ'=á$a_{i,j}$ä，满足$|\varphi'|\geq|\psi|$。

命题2. 若φ =á$a_{i,j}$â和ø=á'$a_{i+k,j}$ó为C的两个C-独立点集，且$|\alpha|\geq|\alpha'|$，则存在C-独立点集φ'=á$a_{i,j}$ä，满足$|\varphi'|\geq|\psi|$。

命题3. 若φ =á$a_{i,j}$â和ø=á'$a_{i,j+k}$ó为C的两个C-独立点集，且$|\alpha|\geq|\alpha'|$，则存在C-独立点集φ'=á$a_{i,j}$ä，满足$|\varphi'|\geq|\psi|$。

由命题1知，在搜索右下行路的过程中，如果已获得了某一C-独立点集φ的某一初始截段á$a_{i,j}$和另一C-独立点集ø的某一初始截段á'$a_{i,j}$，且有$|\alpha'|\leq|\alpha|$，则我们可以停止对ø的进一步搜索。

由命题2知，在搜索右下行路的过程中，在某一列j存在某两个C-独立点集的某初始截段$\varphi=a_{i_1,j_1}\cdots a_{i_s,j}$和$\psi=a_{l_1,m_1}\cdots a_{l_t,j}$，如果$|\varphi|\geq|\psi|$，但$l_t>i_s$，则我们可以停止对ø的进一步搜索。

由命题3知，在搜索右下行路的过程中，在某一行i存在某两个C-独立点集的某初始截段

$\varphi=a_{i_1,j_1}\cdots a_{i,j_s}$和$\phi=a_{i_1,m_1}\cdots a_{i,m_t}$，如果$|\varphi|\geq|\psi|$，但$m_t>j_s$，则我们可以停止对$\phi$的进一步搜索。

由此可见，并不要求搜索所有C的最大C-独立点集，而可以采用比这简单得多的方法进行计算。那么按照我们上面的三个命题，来看如下实例：

i \ j	0(N)	1(E)	2(W)	3(B)	4(O)	5(O)	6(K)	7(S)
0(B)	0	0	0	①	0	0	0	0
1(O)	0	0	0	0	①	1	0	0
2(O)	0	0	0	0	1	①	0	0
3(K)	0	0	0	0	0	0	①	0
4(N)	◈	0	0	0	0	0	0	0
5(E)	0	◈	0	0	0	0	0	0
6(W)	0	0	◈	0	0	0	0	0
7(S)	0	0	0	0	0	0	0	①

首先我们得到φ=B（在φ上的节点用①表示），我们向右下方找路，可以发现，在第4列有两个1，根据命题2，我们选择上面的一个1，也就是说选择第1行的那个1，而不要第2行的那个1。同时我们也发现在第1行也有两个1，由命题3知，我们选择左边的那个1，即第4列的那个1。此时φ=BO。但是当我们的算法运行到第4行时，φ=BOOK，由于K在第3行第6列，而本行的1在第1列，在路φ最后一个节点K的左边，那么我们必须新建一条路ϕ，因为我们并不能确定是否以后就有$|\varphi|\geq|\psi|$，当算法运行到第6行时，φ=BOOK，ϕ=NEW，$|\varphi|$=4，$|\psi|$=3，我们将S链到路φ上，此时我们得到最长右下行路φ=BOOKS，$|\varphi|$=5。这样我们就可以计算出这两个字符串的匹程度。

在我们的算法设计过程中，用到了两个技巧。技巧之一，矩阵C不用存储，是动态建立的，节省了空间。技巧之二，本算法并不要求所有的S与T中所有的元素都相互进行比较，也并不存储所有的右下行路，节省了时间和空间。由矩阵中1的出现情况可见，本算法所需的空间和时间都远小于O（mn）

4 结束语

本文给出了一个与模式匹配不同的，具有若干应用的，串的最大匹配算法，该算法已经机器上实现，达到了预期的效果。本文仅讨论权值恒为1的情况，对于权值任意的情形不难由此得到推广。

参 考 文 献

[1] A. Apostolico. String editing and longest common subsequences. In G. Rozenberg and A. Salomaa, editors, Handbook of Formal Languages, volume 2 Linear Modeling: Background and Application, chapter 8, pages 361-398. Springer-Verlag, Berlin, 1997.

[2] A. Apostolico and Z. Galil, editors. Pattern matching algorithms. Oxford University Press, 1997.

[3] D.Breslauer,Saving Comparisions in the Crochemore-Perrin String Matching Algorithms. In proc. of 1st European Symp. On Algorithms, pp. 61-72, 1993.

[4] M. Crochemore, A. Czumaj, L. Gasieniec, S. Jarominek, T. Lecroq, W. Plandowski, and W. Rytter. Speeding up two string matching algorithms, Algorithmaica 12(4/5)(1994), pp. 247-267.
[5] M. Crochemore and D. Perrin, Two-way string-matching. J. Assoc. Comput. Mach., 38(3)(1991), pp. 651-675.
[6]. Z. Galil and J. Seiferas, Time-space-optimal string matching. J.Comput. System Sci., 26(3) (1983), pp280-294.
[7] Z. Galil, On improving the worst case running time of the Boyer-Moore string searching algorithm. CAXM 22(9)(1979), pp505-508.
[8] L.Gasieniec, W. Plandowski and W. Rytter, The zooming method: a recusive approach to time-space efficient string-matching. Theoret. Comput. Sci., 147(1/2)(1995), pp.19-30.
[9] D.E.Knuth, J.H.Morris and V.R.Pratt, Fast pattern matching in strings, SIAM J. Comput., 6(1) (1977), pp. 322-350.
[10] A.C.Yao, The Complexity of Pattern Matching for a Random String, SIAM Journal on Computing, 8(3)(1979), pp.368-387.
[11] D. Sankoff and J. B. Kruskal. Time warps, string edits, and macromolecules: the theory and practice of sequence comparison. Addison-Wesley, Reading, MA, 1983.

作者：
林毓材，男，教授，主要研究领域为计算机基础理论及应用等。
向永红，男，硕士，主要研究方向为图论、并行计算。
张春霞，女，硕士，主要研究方向为MAS（多Agent系统）。
张建军，男，硕士，主要研究方向为图像处理。
本文通讯联系人：向永红，云南师范大学计算机科学系97研，
e-mail：xyoho@ynmail.com

◇目录◇上一篇◇下一篇◇