文章编号: 1001- 4764(2002) 02- 0078- 03

字符串的模式匹配算法

—— 基于 KMP算法的讨论

李 静

(青岛化工学院信息与控制工程学院, 山东 青岛 266042)

摘 要: 重点对基本的串匹配算法和 KMP算法进行了探讨。通过对这两种算法的比较分析提出了一个新算法,此算法具有比基本的串匹配算法更优越的时间复杂性,并且相对 KMP算法而言更简洁易懂。

关键词: 子串; 目标串; 模式串; 模式匹配; KMP算法; 无回溯

中图分类号: TP 311. 12 文献标识码: A

The Arithmetic of Matching the String's Mode

—— Talking Over the KMP Arithmetic

LI Jing

(College of Information and Control Engineering, Qingdao Institute of Chemical Technology, Shandong Qingdao 266042)

Abstract The basic arithmetic of matching the string's mode and KMP arithmetic are discussed in this paper. The new arithmetic was put forward through the analysis for the two arithmetic. This arithmetic has some advantage of less time complexity and more simple than KMP arithmetic.

Key words sub-string; target-string; mode-string; matching the string's mode; the KMP arithmetic; non-recounting

一般使用的计算机的硬件结构主要反映数值计算的需要,而计算机上的非数值处理的对象基本上是字符串数据,因此在处理字符串数据时比处理整数和浮点数要复杂得多。随着程序设计语言加工程序的发展,字符串的处理也有了越来越多的研究。子串的定位操作通常称为串的模式匹配,是各种串处理系统中最重要的操作之一。基本的串匹配算法是一种传统的算法,而 KMP算法则是一种无回溯的算法,时间复杂性在有些情况下较好,但算法本身不易理解。本文提出一种对 KMP算法的一种简约算法,此算法简洁明了,其时间复杂性也较好。

1 串模式匹配算法

1.1 基本的串匹配算法

两个串: S串、T串(T串不空)。若串 S中存在和 T相等的子串(若干连续的字符组成的子序列),称匹配成功,否则,称匹配不成功。一般称串 S为目标串,串 T为模式串。

算法的基本思想是: 从目标串 S的第一个字符起和模式串 T的第一个字符比较,若相等则继续逐个比较后继字符,否则从模式串 S的第二个字符起再重新和模式串 T的第一个字符比较。依次类推,直至模式 T中的每个字符依次和目标 S

收稿日期: 2001- 05- 11
作者简介: 李 静(1971-),女,讲师,青岛海洋大学在职硕士研究生

中的一个连续的字符序列相等,则匹配成功,定位是和模式 T中第一个字符相等的字符在目标 S中的位置;否则匹配不成功。以下是模式 T='ABCBA'和目标 S的匹配过程。

第一趟匹配 S: A B A B C A B C A C B A B
= = ≠
T: A B C A C

第二趟匹配 S: A B A B C A B C A C B A B
≠
T: A B C A C

第三趟匹配 S: A B A B C A B C A C B A B
= = = = ≠
T: A B C A C

第四趟匹配 S: A B A B C A B C A C B A B
≠
T: A B C A C

第五趟匹配 S: A B A B C A B C A C B A B
≠
T: A B C A C

第六趟匹配 S: A B A B C A B C A C B A B
= = = = =
T: A B C A C

该算法的匹配过程易于理解,在文本编辑等场合效率较高,算法的时间复杂性为 O(n+ m)(注: n和 m分别为目标串和模式的长度)。然而,在有些情况下,该算法的效率却很低,时间复杂度为 O(m* n)。一个较好的模式匹配算法是 KMP算法。

1.2 KMP算法(一种改进的模式匹配算法)

该算法的时间复杂度为 O(m+ n)。其改进在于:每当一趟匹配过程中出现字符不等时,不需回溯目标串,而是由已经得到的"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。以下是使用 KMP算法的 S串和 T串的匹配过程。

第一趟匹配 S: A B A B C A B C A C B A B
= = ≠
T: A B C A C

第二趟匹配 S: A B A B C A B C A C B A B
= = = = ≠
T: A B C A C

第三趟匹配 S: A B A B C A B C A C B A B
= = = = =
T: A B C A C

当第一趟匹配中目标串中第 3个字符(位置以下用 i标识)和模式串中第 3个字符(位置以下用 j标识)不相等时,i不变,模式串向右"滑动",即从模式串第 j个字符之前查找一个字符代替第 j个字符和目标串的第 i个字符继续比较。在第二趟匹配中就是用第 1个字符代替第 j个字符与目标串的第 i个字符进行比较。

改进后匹配算法的关键问题是:匹配过程中当字符不相等时,在模式串中找哪个字符代替第 j个字符和目标串中的第 i个字符比较,即模式串"向右滑动"多远?假设此时应是模式中的第 k(k <j)个字符与目标串中的第 i个字符比较,则模式中前 k- 1个字符的子串必须满足下列关系式(1),且不可能存在 k> k满足关系式(1)

$$'t_1t_2\ldots t_{k-1}' = 'S_{i-k+1}S_{i-k+2}\ldots S_{i-1}' \quad (1)$$

而匹配过程中已经得到的"部分匹配"结果是:

$$'t_1t_2\ldots t_{j-k+1}t_{j-k+2}\ldots t_{j-1}' = 'S_{i-j+1}S_{i-j+2}\ldots S_{i-k+1}S_{i-k+2}\ldots S_{i-1}' \quad (2)$$

由(1)和(2)推得下列等式:

$$'t_1t_2\ldots t_{k-1}' = 't_{j-k+1}t_{j-k+2}\ldots t_{j-1}' \quad (3)$$

若模式串中存在满足式(3)的两个子串,则当匹配过程中,主串中第 i个字符与模式中第 j个字符不等时,仅需将模式向右滑动至模式中第 k个字符和主串中第 i个字符对齐,此时模式中前 k- 1个字符的子串'$t_1t_2\ldots t_{k-1}$'必定与主串中第 i个字符之前长度为 k- 1的子串'$S_{i-k+1}S_{i-k+2}\ldots S_{i-1}$'相等(由上面的推导可得),由此匹配仅需从模式中第 k个字符与主串中第 i个字符比较起继续进行。

若令 next[j]= k,则 next[j]表明当模式中第 j个字符与主串中相应字符不相等时,在模式中需重新和主串中该字符进行比较的字符的位置,这一位置只与模式串本身有关而与主串中的字符无关。由此模式串的 next函数的定义为:

$$next[j]=\begin{cases}0 & \text{当 } j=1\text{时}\\ Max\{k \mid 1<k<j\text{且}'t_1t_2\cdots t_{k-1}'='t_{j-k+1}t_{j-k+2}\cdots t_{j-1}'\} & \text{当此集合不空时}\\ 1 & \text{其它情况}\end{cases}$$

则 KMP算法在形式上与基本模式匹配算法相似,不同之处仅在于:当匹配过程中比较不相等时,主串中 i不变,模式中 j退回到 next[j]所指示的位置,并且当指针 j退到 0时,i和 j都加 1,即模式串的第 1个字符和主串的第 i+ 1个字符进

行比较。

说明: 基本的串匹配算法的时间复杂性是 O(n* m),但在一般情况下,其实际的执行时间近似于 O(n+ m),因此至今仍被采用。而 KMP算法仅当模式与主串之间存在许多“部分匹配” (即每趟比较若干字符后才发现不匹配)的情况下时才显示出其时间优越性,且其最大特点是指示主串的指针不需回溯——从头至尾扫描一遍,这对处理从外部设备输入的庞大文件很有效,无需重读,可以边读边匹配。

1.3 对 KMP匹配算法的简约

在一般情况下,基本的串匹配算法的时间复杂性近似于 O(n+ m),此时 KMP算法的优势并不明显,而算法本身又有些难理解,分析其复杂性,在此提出一种简约算法。

算法的基本思想是: 目标串 S和模式串 T(T串不空)进行匹配,i j都为 1,表示从 S串和 T串的第 1个字符开始比较,若相等,则 i和 j都加 1,表示对下一个字符继续比较,此时还要比较当前第 i个字符是否和模式串的第 1个字符第 1次相等 (第 1次相等用一状态变量标识),如若是第 1次相等用 k记住其在目标串中的位置,并标记为不是第 1次相等 (下一字符再与 T串的第 1个字符相等时由于已不是第 1次相等了,所以 k不再更新);若 S串的第 i个字符和 T串的第 j个字符不相等,且 k的值不为 0,即已标记为某个和 T串的第 1个字符相等的 S串中的字符的有效位置,则只需将 j更新为 2,将 i更新为 k+ 1,即将 T串的第 2个字符和 S串的第 k+ 1个字符进行比较,因为在这趟比较中此时 k的值是 S串的起始比较字符后面的几个字符中与 T串的第 1个字符第 1次相等的字符的位置。以下是使用简约匹配算法的 S串和 T串的匹配过程。

第一趟匹配 S: A B ABCABC ACBAB
= =≠
T: A B C A C (k= 3)

第二趟匹配 S: ABA B C A BC ACBAB
= = =≠
T: A B C A C(k= 6)

第三趟匹配 S: ABABC A B C ACBAB
= = = = =
T: A B C A C

在第一趟匹配过程中: k为 0,i j从都为 1开始比较 S串的第 1个字符和 T串的字符,若相等,则分别加 1后,再比较两串的第 2个字符,同时看当前 S串的第 i(现 i= 2)个字符与 T串的第 1个字符是否相等 (现第 1次相等的状态为‘真’),若相等,给 k赋值,现不相等,所以继续比较,i和 j分别加 1,即比较两串的第 3个字符。不相等,在结束本趟匹配前,还要比较此时 S串的第 i(现 i= 3)个字符与 T串的第 1个字符是否相等 (现第 1次相等的状态为‘真’),此时相等,则给 k赋值为 3,并标记第 1次相等的状态为‘假’,即以后再有相同情况将不再给 k赋新值。

在第二趟匹配过程中: 初始第 1次相等的状态为‘真’,k为 0, j为 2, i为 k+ 1,即 S串从第 4个字符开始和 T串的第 2个字符比较,同时比较 S串的第 i个字符与 T串的第 1个字符是否第 1次相等,若相等则给 k赋 i的值且标记第 1次相等的状态为‘假’,若 S串、T串的第 i和 j位置对应字符相等,则分别加 1继续比较,否则结束本趟匹配,如上例。

1.4 对 KMP算法的简约算法浅析

KMP算法的简约算法,在一般情况下其时间复杂性也近似于 O(n+ m),且在有些情况下其时间复杂度比基本的模式匹配算法要好。KMP算法中 next[j]的计算较难于理解,而此算法只需在比较过程中,在第 1次相等状态为‘真’的情况下增加了一次 S串中当前比较字符和 T串的第 1个字符的比较,若相等则把 S串的当前位置用一变量记住,且把第 1次相等的状态置为‘假’,即在本趟的继续比较中不再需要额外的这一次比较了, S串和 T串继续下一字符的比较。该算法已用具体的高级语言实现 (例: C语言、PASCAL语言等)。同时此算法在比较过程中产生回溯,所以不存在 KMP算法在处理从外设输入的庞大文件时的有效的无回溯的特点。

参 考 文 献

[1] 严蔚敏,吴伟民.数据结构 [M].北京: 清华大学出版社, 1999. 6

[2] William Topp.数据结构: C+ + 语言描述 [M].北京: 清华大学出版社 (译), 1997

[3] Gonnet G H. Handbook of Algorithms And Data Structure [M]. Addison-Wesley Publishing Company, 1999

[4] 马建,腾弘飞,孙治国等.图形匹配问题 [J].计算机科学, 2001, 28(4): 61