一种模式匹配快速算法

刘玉龙[1] 刘 啸[2]

(徐州师范大学计算机学院[1] 徐州师范大学现代教育技术中心 徐州 221116)[2]

摘 要 在定义模式串的特征值之后,给出了判断两等长串匹配的必要条件以及两相邻子串的特征值之间的递推关系。在此基础上,提供一种模式匹配快速算法,其时间复杂度可达 $O(n)$。该算法彻底避免了回溯现象,执行效率要比 RK 算法高。

关键词 串,模式,模式匹配,特征值,递推

A Fast String-matching Algorithm

LIU Yu-Long[1] LIU Xiao[2]

(College of Computer[1], Center of Modern Education Technology[2], Xuzhou Normal University, Xuzhou 221116)

Abstract After introducing the definition of a string's eigenvalue, necessary conditions used to judge if two strings having equal lengths are matched, and a recurrence relation between the eigenvalues of two adjacent substrings are given. Then a fast string-matching algorithm which expected running time is $O(n)$ is presented. The algorithm is more effective than RK algorithm and avoiding all backtrackings thoroughly.

Keywords String, Pattern, String-matching, Eigenvalue, Recursion

1 引言

对于给定的长度为 l 的模式串 T,需要确定 T 在另一长度为 n(一般 $l<<n$)的串 S(也称其为母串)中是否出现以及出现的位置,这就是模式匹配,也称模式识别,又称串匹配或串查找,它是最常用的计算机基本操作之一。

模式匹配问题可以分解为模式串与母串的所有长度为 l 的子串的匹配问题,因此模式匹配算法的核心是如何能够尽快判断出两个长度均为 l 的串是否相等。在相等的情况下,各种算法均需至少比较 l 次,效率方面不可能再提高;而在不相等时,各种算法的效率则可能相差甚远。因此,模式匹配算法的改进,着眼点应放在如何以最小代价判断出两等长的串是否不等方面。为此,首先引入串的特征值的概念,然后给出两串相等的必要条件。

定义 1(串的特征值) 给定串 S,串 S 的特征值 v_S 为 S 中各字符的编码之和。

定理 1 在无重码编码系统中,两串匹配的必要条件是:

①两串长度相等;

②两串特征值相等。

证明:①显然成立,用反证法证②。设两串特征值不等,则两串至少有一对字符的编码不等,即至少有一对字符不等,从而两串不匹配。

2 提高模式匹配算法性能的基本思路

为了提高模式匹配算法的性能,我们对母串 S 的结构做进一步的分析。如前,设母串 S 的长度为 n,模式串 T 的长度为 l ($l<<n$),则有

(1)需进行模式匹配判断的子串 S_k 的个数为 $n-l+1$,其中能与 T 匹配的子串(设为 m 个)一般占极少数,与 T 不匹配的个数占绝大多数。

(2)模式匹配算法的实质是需同时完成如下两项工作:

①找出所有与 T 匹配的子串 S_k;

②找出所有与 T 不匹配的子串 S_k。

另外,前面我们已经指出,关于与 T 匹配的子串 S_k 的判定效率本质上不可能再改进,因而提高不匹配子串 S_k 的判定效率,则可显著提高整体匹配算法的性能。于是我们有

(3)算法改进的重点应放在以尽可能少的代价找出所有与 T 不匹配的子串 S_k。

(4)根据两串匹配的必要条件(定理 1),若能以较小代价得到子串 S_k 的特征值,只要与 T 的特征值进行比较,即可判断 S_k 是否与 T 不匹配,因此问题转化为如何以最小代价求出 S_k 的特征值。

(5)若能找出两相邻子串 S_k 和 S_{k+1} 之间的联系,则可能以较小代价依次求出所有长度为 l 的子串的特征值。为此,设子串 $S_k=s_k s_{k+1}\cdots s_{k+l-1}$,子串 $S_{k+1}=s_{k+1}\cdots s_{k+l-1}s_{k+l}$,则如图 1 所示,$S_{k+1}$ 可由 S_k 删去首字符 s_k 再在尾端联接字符 s_{k+l} 后得到。

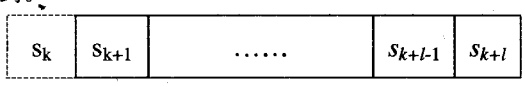

图 1 相邻子串间的关系

(6)于是两相邻子串 S_k 和 S_{k+1} 的相应特征值 $v_{s_{k+1}}$ 与 v_{s_k} 之间具有如下关系:

$$v_{s_{k+1}}=v_{s_k}-\mathrm{code}(s_k)+\mathrm{code}(s_{k+l})$$

其中,code(c)代表字符 c 的某种编码。

(7)在特征值 v_s 与模式串 T 的特征值 v_T 相等的所有等长子串 s_k 中,当模式串中有相同字符时可有少数几种与模式串 T 匹配(情况 1),而当模式串中没有相同字符时只有一种

刘玉龙 教授,主要研究方向为算法研究和信息安全;**刘 啸** 硕士,主要研究方向为数据库和信息安全。

与模式串 T 匹配(情况 2),故模式匹配算法在处理特征值与模式串特征值相等的子串时,应以尽可能小的代价将与 T 不匹配的子串判断出来。

(8)对于(7)中的情况 2,只需比较 1 次即可完成判断。

(9)对于其它情况,可采用三点(即首尾中)匹配算法:首先检测首字符是否匹配,若匹配再检测尾字符是否匹配,若匹配再检测中点字符。如此处理的目的是扩大探测范围,尽快检测出不匹配子串 s_k。

3 模式匹配快速算法

基于上述讨论,给出判断长度为 l、特征值相等的子串 S_k 是否与模式串 T 匹配的三点匹配算法和关于母串 S 的模式匹配快速算法。

算法 1(三点匹配算法)

(1)判断首字符 t_1 和 s_k 是否相等。若相等,转(2)继续,否则两串 S_k 与 T 不匹配,做不匹配标记,算法结束。

(2)判断尾字符 t_l 和 s_{k+l-1} 是否相等。若相等,转(3)继续,否则两串 S_k 与 T 不匹配,做不匹配标记,算法结束。

(3)判断中点字符 $t_{\lfloor (l+1)/2 \rfloor}$ 和 $s_{\lfloor k+(l-1)/2 \rfloor}$ 是否相等。若相等,转(4)继续,否则两串 S_k 与 T 不匹配,做不匹配标记,算法结束。

(4)用朴素匹配算法判断左子串 $t_2t_3\cdots t_{\lfloor (l+1)/2 \rfloor -1}$ 和左子串 $s_{k+1}s_{k+2}\cdots s_{\lfloor k+(l-1)/2 \rfloor -1}$ 是否匹配。若匹配,转(5)继续,否则两串 S_k 与 T 不匹配,做不匹配标记,算法结束。

(5)用朴素匹配算法判断右子串 $t_{\lfloor (l+1)/2 \rfloor +1}t_{\lfloor (l+1)/2 \rfloor +2}\cdots t_{l-1}$ 和右子串 $s_{\lfloor k+(l-1)/2 \rfloor}s_{\lfloor k+(l-1)/2 \rfloor +2}\cdots s_{k+1-2}$ 是否匹配。若匹配,两串 S_k 与 T 匹配,做匹配标记,算法结束;否则两串 S_k 与 T 不匹配,做不匹配标记,算法结束。

算法 2(模式匹配快速算法)

//设长度为 $n(l<<n)$ 的母串 S 和长度为 l 的模式串 T 分别存放于数组 $S[1:n]$ 和 $T[1:l]$ 中。

(1)求模式串 T 的特征值:$v_T=\sum_{i=1}^{l}\mathrm{code}(T[i])$,求母串 S 的第一个长度为 l 的子串 S_1 的特征值:$v_s=\sum_{i=1}^{l}\mathrm{code}(S[i])$;

(2)for($k:=1;k<=n-l;k++$)//依次判定当前子串 S_k 是否匹配;

```
{
  if(v_s! =v_T)//不满足匹配的必要条件,不匹配,右移一位,计算新子串的特征值
    {
      v_s=v_s-code(S[k])+code(S[k+l])
    }
  else//特征值相等,进一步判定当前子串是否匹配
    {
      调用三点匹配算法
      if(不匹配)
        {
          v_S=v_S-code(S[k])+code(S[k+l])
        }
      else
        {
        输出 k
          v_S=v_S-code(S[k])+code(S[k+l])
      }
  }
```

```
}
(3)if (v_s=v_T)//判断子串 S_{n-l+1} 是否与模式串 T 匹配
    {调用三点匹配算法
      if(匹配)
        输出 n-l+1
    }
(4)算法结束
```

4 算法效率分析

在算法的第(1)步中求模式串 T 的特征值 v_T 和子串 S_1 的特征值 v_s 共需 $2l$ 次加运算,第(2)步中求子串 S_k 的特征值 v_s 共需 $2(n-l)$ 次加减运算,因此算法 2 共需 $2n$ 次加减运算。

算法判断各子串 S_k 的特征值 v_s 与模式串 T 的特征值 v_T 是否相等,共需 $n-l+1$ 次比较运算。

设母串中共有 $m(0\leqslant m\leqslant n-l+1)$ 个子串 S_k 其 v_s 与 v_T 相等,且其中有 $m'(m'\leqslant m)$ 个子串 S_k 与 T 匹配,则其中与 T 不匹配的子串有 $m-m'$ 个,将它们区分开,需进一步判断。分两种情况:模式串中的字符各异和模式串中有相同字符。

对于第一种情况,区分不匹配子串仅需 $m-m'$ 次比较运算,而区分匹配子串共需 $m'l$ 次比较运算,对这 m 个子串的处理共需 $m-m'+m'l$ 次比较,算法 2 共需 $n-l+1+m-m'+m'l$ 即 $n+(m'-1)l+(m-m')+1$ 次比较运算。

对于第二种情况,由于相同字符的个数及分布情况比较复杂,只能考虑最坏情况,此时对这 m 个子串的处理最多需 ml 次比较运算,因此算法 2 最多需 $n-l+1+ml$ 即 $n+(m-1)l+1$ 次比较运算。

当模式串 T 的字符各异且母串 S 中无与 T 匹配的子串时,总比较次数变成 $n-l+m+1$。

在最坏情况下(即 $m=n-l+1$),比较次数为 $nl+n-l^2+1$。

一般情况下,由于 $m'<<m<<n$ 及 $l<<n$,可以得到算法 2 的时间复杂度为 $O(n)$。

结束语 KMP 算法对模式串中有重复字符的匹配问题可以避免回溯现象,但对于字符各异的模式串的匹配问题则无能为力,而本算法则彻底避免了回溯现象。与 RK 算法比较,本算法既避免了因散列函数的选择不当可能引起的算法效率下降,而且本文提供的求特征值的递推公式的计算工作量($2n$ 次加减运算)要比 RK 算法提供的求散列函数值的递推公式的计算工作量($2n$ 次加减运算、$2n$ 次乘运算、$n+l$ 次 mod 运算)要小得多,从而执行效率要比 RK 算法高。当模式串 T 字符各异、匹配子串 S_k 数量特别少时,本算法特别有效。

参 考 文 献

1 Cormen T H, Leiserson C E, Rivest R L. Clifford Stein. In: Introduction to algorithms. Second Edition. Cambridge: The MIT Press, 2001

2 王晓东. 计算机算法设计与分析. 第 2 版. 北京:电子工业出版社,2004

3 周培德. 算法设计与分析. 北京:机械工业出版社,1998