**对ZZL字符串匹配算法的改进**

**——ZZL最短匹配定理  
张亮**

     ZZL算法是一种可做特殊用途的字符串匹配算法，本文将改进ZZL算法，在ZZL算法的基础上，提出了ZZL最短匹配定理，根据模式串的自身特征以进一步减少不必要的匹配次数。

**字符串匹配**

     字符串匹配的含义就是：在主串S中，从位置start开始查找是否存在模式串（也称作模式串）T，如在主串S中查找到一个与模式串T相同的模式串，则模式串与主串匹配；如在主串S中未查找到一个与模式串T相同的模式串，则不匹配。字符串匹配用途广泛，人们很早就对字符串匹配算法进行了研究，在BF（Brute Force）算法的基础上，提出了一些优秀的串匹配算法，比如经典的KMP匹配算法，BM算法等。  
     首先全文都有如下假设：  
匹配的主串为S：S[1…N]，长度为N；模式串为T：T[1…M]，长度为M；N≥M；

**ZZL匹配算法**

**匹配思想**

     现有的字符串匹配算法不论是按照模式串从左至右还是从右至左的顺序匹配，都是直接进行比较，而ZZL算法 的核心思想是：首先在主串S中查找模式串T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，从这些位置开始继续匹配模式串T。对于频繁使用 的要匹配的主串和模式串来说，由于预先保存了模式串在主串中的所有存储位置，所以匹配速度会非常快。

**处理过程**

     ZZL算法也分为预处理和匹配两个阶段，预处理主要完成查找模式串首字符在主串中的所有出现位置，并将其保存在一个数组中；在预处理的基础上，字符串匹配算法就可以从查找到的模式串在主串中的位置开始，采用BF匹配模式串首字母之后的其余部分。  
时间按复杂度  
     ZZL算法的特点是简单容易实现，如果不考虑算法的预处理过程，若模式串首字母在主串中出现k次，则ZZL算法最坏情况下比较次数为k\*(M-1)<k\*M。如果考虑算法的预处理过程，则总的比较次数需再加上N次，即为k\*M+N。  
ZZL最短匹配定理  
ZZL缺点分析  
     观察ZZL算法，如果模式串首字母在主串中出现次数很多，那么ZZL算法记录的匹配点将会增加，时间复杂度也随之增加。让我们先来考察下面的匹配场景：  
S = to**m** orrow, **m**any and **m**any **m** anufactories will close.  
T = manufactories  
     那么ZZL算法在预处理S时会找到4个开始匹配点，在上面的S串中用**红色** 标明，然后从每个匹配点开始采用BF算法尝试匹配。然而事实上，只有在最后一个匹配点上T和S才匹配成功。前面的3个匹配点都不能使T和S匹配，如果能在预处理时就尽量排除这样的点岂不是效率更高吗？

**ZZL最短匹配定理**

     ZZL算法并没有利用模式串T的特征，然而模式串T的特征能够帮助我们有效的减少匹配点。  
     考察模式串T，我们发现T的首字符m仅出现了一次，而在S串中第二个m和第一个m的间隔小于T的长度，因此第一个m绝对不可能是正确的匹配点，可以直接排除。事实上，这是下面定理的一个特殊情况。  
     首先定义模式串T的一个性质。  
**模式串T的最短ZZL匹配间距：**   
对任意模式串T，设T的首字符为c，c在T中出现的第二次位置为pos，如果c在T中仅出现一次，那么令pos=M；  
位置pos就是T的最短ZZL匹配间距，使用T.W来表示，易得T.W <= M；  
**ZZL最短匹配定理：**

对任意主串S和模式串T，设T的首字符c在S中的连续出现位置为u和v，v >u。如果v – u < T.W，那么在u位置S和T必定不匹配。  
证明：  
假设v – u < T.W，假设在u位置S和T匹配，那么有  
S[u] = T[0] = c;  
S[u+1] = T[1];  
…  
S[u+M-1] = T[M-1];  
  
由假设v – u < T.W  
因此v必定位于区间(u, u+T.W)之中，并且T.W <= M，于是可得到  
S[v] = T[i]，0< i < T.W <= M ------ (1)  
而v是T的首字符c在S中的出现位置，于是有  
S[v] = c = T[0] ------ (2)  
根据上面(1)和(2)两式有T[i] = T[0], 0< i < T.W，这将违反T.W的定义。因此当v – u < T.W时，在u位置S和T必定不匹配。  
证毕。

**改进的ZZL算法**

     把ZZL最短匹配定理应用于ZZL算法就得到了改进后的ZZ算法，改进点是在ZZL的预处理阶段，对匹配阶段无影响。  
     预处理阶段分为两部分，第一部分是求出T的ZZL最短匹配间距，第二阶段是求出T在S中的匹配点。  
     求T的ZZL最短匹配间距的伪代码如下所示：  
  
c = T[0]  
T.W = M  
for i = 1 to M  
begin  
if T[i] == c then  
    T.W = i  
break;  
        end if  
    end  
求T在S中的匹配点的伪代码如下所示：  
j = 0;  
for i = 0 to N - M  
begin  
      if S[i] == T[0] then // a potential match position  
            if j == 0 || i – next[j-1] >= T.W then   
                 next[j] = i;  
                 j = j + 1;  
            else  
                 next[j-1] = i; // discard the previous one  
            end if  
      end if  
end

     如果不考虑算法的预处理过程，若模式串T首字母在主串S中出现k次，且满足ZZL最短匹配间距，则ZZL算法最坏情况下比较次数为k\*(M-1)。如果考虑算法的预处理过程，则总的比较次数为k\*(M-1) + M + (N-M) = k\*M + N。  
因此对比ZZL算法，改进后的ZZL算法将能剔除那些T的首字母在S中不能满足ZZL最短匹配间距的出现，从而减少比较次数和提高匹配速度。

**结论**

改进的ZZL算法利用模式串T本身的特征，通过ZZL最小匹配定理，能够有效的减少ZZL的比较次数，提高匹配速度。  
**参考文献**   
1.    朱战立编著. 数据结构——使用C语言（第3版）[M]. 西安：西安交通大学出版社，2004  
2.    纪福全 朱战立. 一种可做特殊用途的字符串匹配算法 计算机与信息技术[J] 200608