微视频

KMP串匹配算法

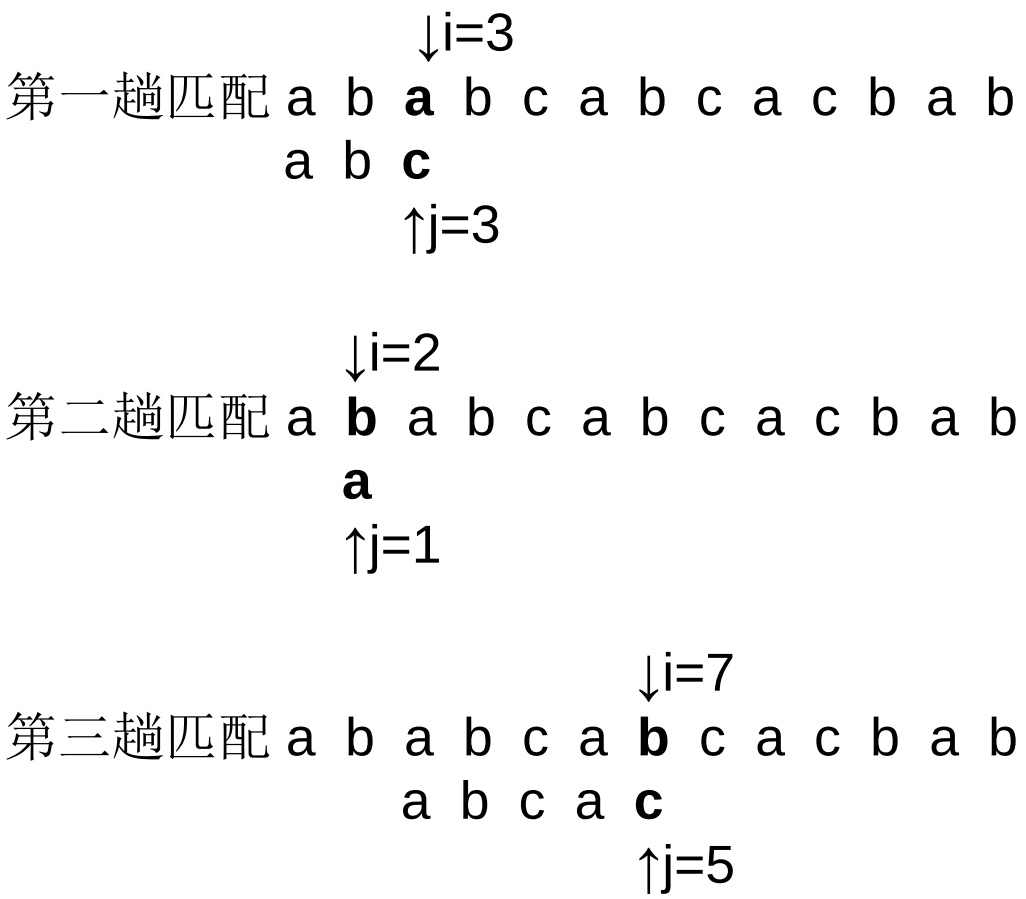
中国海洋大学信息学院
魏振钢
Tel:0532-66781226
Email:wzgwzq@ouc.edu.cn

传统的模式匹配算法

```
int Index(SString S, SString T, int pos) {
   // 返回子串T在主串S中第pos个字符之后的位置。若不存在，
    // 则函数值为0。其中，T非空，1≤pos≤StrLength(S)。
    i = pos;   j = 1;
    while (i <= S[0] && j <= T[0]) {
      if (S[i] == T[j]) { ++i;  ++j; }   // 继续比较后继字符
      else { i = i-j+2;   j = 1; }     // 指针后退重新开始匹配
    }
   if (j > T[0])  return  i-T[0];
   else return 0;
} // Index                                          算法4.5中
```

算法的基本思想是：从主串S的第 pos 个字符起和模式的第一个字符比较之，若相等，则继续逐个比较后续字符；否则从主串的下一个字符起再重新和模式的字符比较之。依此类推，直至模式 T 中的每个字符依次和主串 S 中的一个连续的字符序列相等，则称**匹配成功**，函数值为和模式 T 中第一个字符相等的字符在主串 S 中的序号，否则称**匹配不成功**，函数值为零。

例如，主串为‘a b a b c a b c a c b a b’ ，模式为‘abcac’



例如，主串为‘a b a b c a b c a c b a b’，模式为‘abcac’

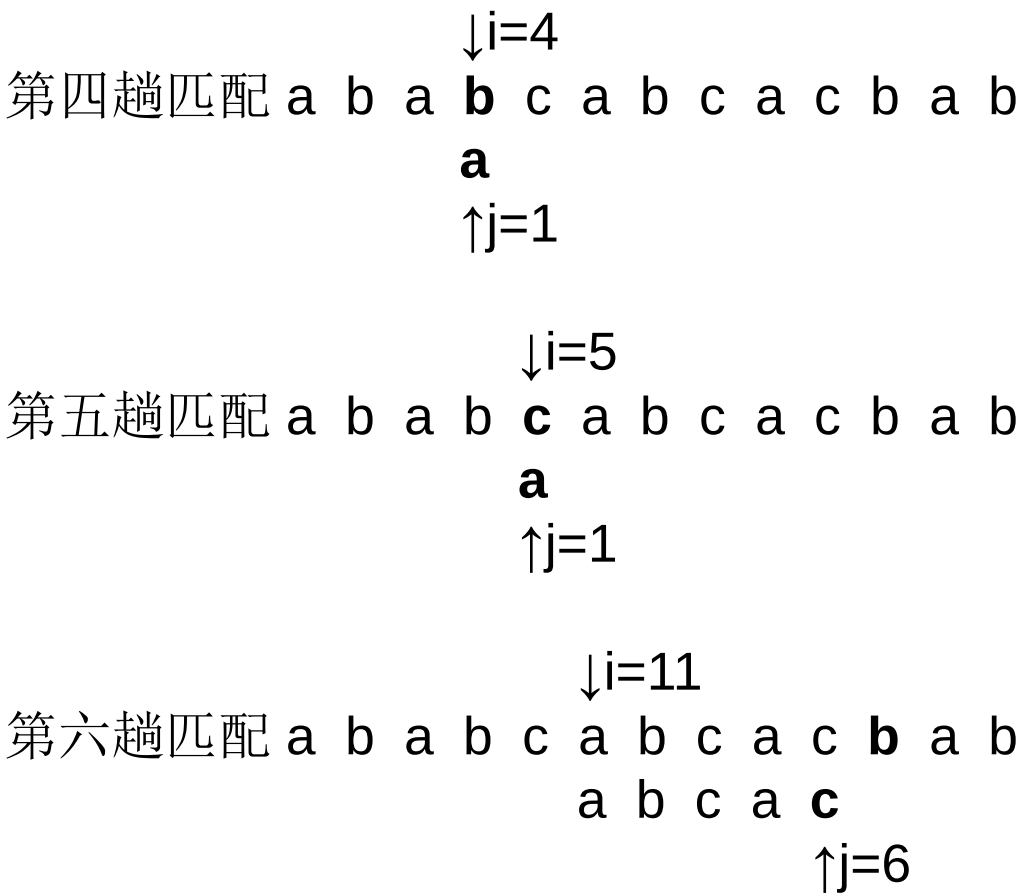


图4.3 算法4.5的匹配过程

回顾图4.3中的匹配过程，在第三趟的匹配中，当i=7、j=5字符比较不等时，又从i=4、j=1重新开始比较。然后，经仔细观察可发现，在i=4和j=1，i=5和j=1以及i=6和j=1这3次比较都是不必进行的。因为从第三趟部分匹配的结果就可得出，主串中第4、5和6个字符必然是‘b’、‘c’和‘a’（即模式串中第2、3和4个字符）。因为模式中的第一个字符是a，因此它无需再和这3个字符进行比较，而仅需将模式向右滑动3个字符的位置继续进行i=7、j=2时的字符比较即可。同理，在第一趟匹配中出现字符不等时，仅需将模式向右移动两个字符的位置继续进行i=3、j=1时的字符比较。由此，在整个匹配的过程中，i指针没有回溯，如图4.4所示。

改进KMP算法：每一趟匹配过程中出现字符比较不等时，不需回溯i的指针，而是利用已经得到的部分匹配结果将模式向右‘滑动’尽可能远的一段距离后，继续进行比较。

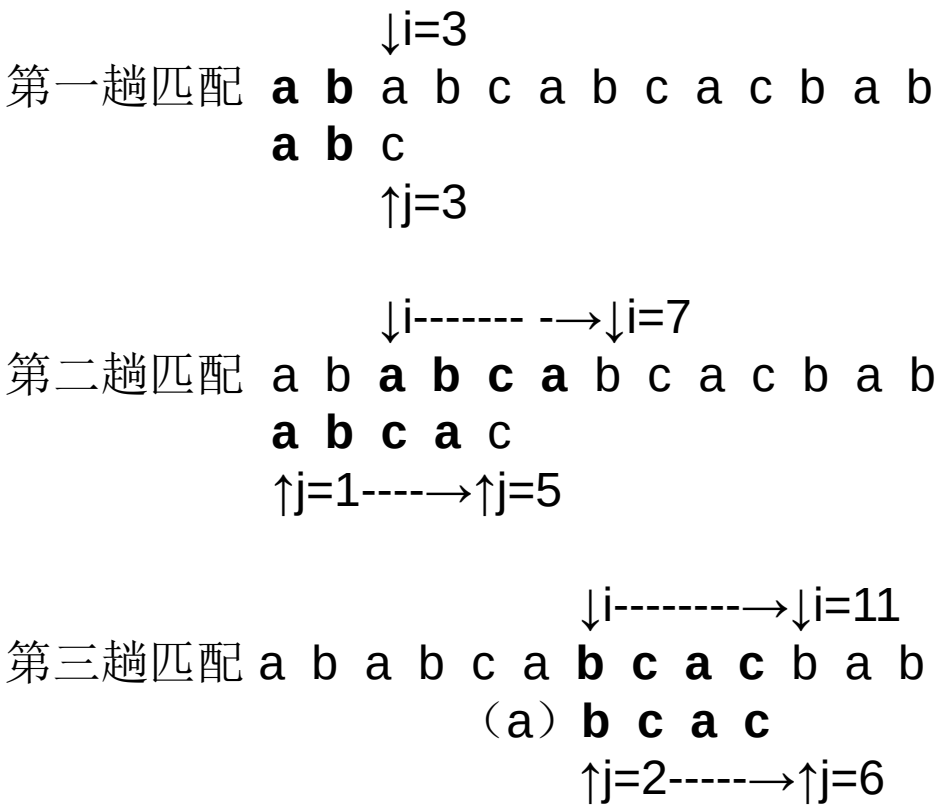


图4.4 改进算法的匹配过程示例

现讨论一般情况，假设主串为‘$s_1s_2...s_n$’,模式串为‘$p_1p_2...p_m$’,为了实现改进算法，需要解决下述问题：当匹配过程中产生‘失配’（即$s_i \neq p_j$）时，模式串‘向右滑动’可行的距离多远，即，当主串中第 i 个字符与模式中第 j 个字符‘失配’时，主串中第 i 个字符（i 指针不回溯）应与模式中哪个字符再比较？

假设此时应与模式中第 k（k<j）个字符继续比较，则模式中前 k-1个字符必须满足下列关系式(4-2)，且不可能存在k′>k满足下列关系式(4-2)

‘$p_1p_2...p_{k-1}$’ =’ $s_{i-k+1}s_{i-k+2}...s_{i-1}$’ (4-2)

而已经得到的‘部分匹配’的结果是

$$‘p_{j-k+1}p_{j-k+2}\cdots p_{j-1}’ = ’ s_{i-k+1}s_{i-k+2}\cdots s_{i-1}’ \qquad (4\text{-}3)$$

由式（4-2）和式（4-3）推的下列等式

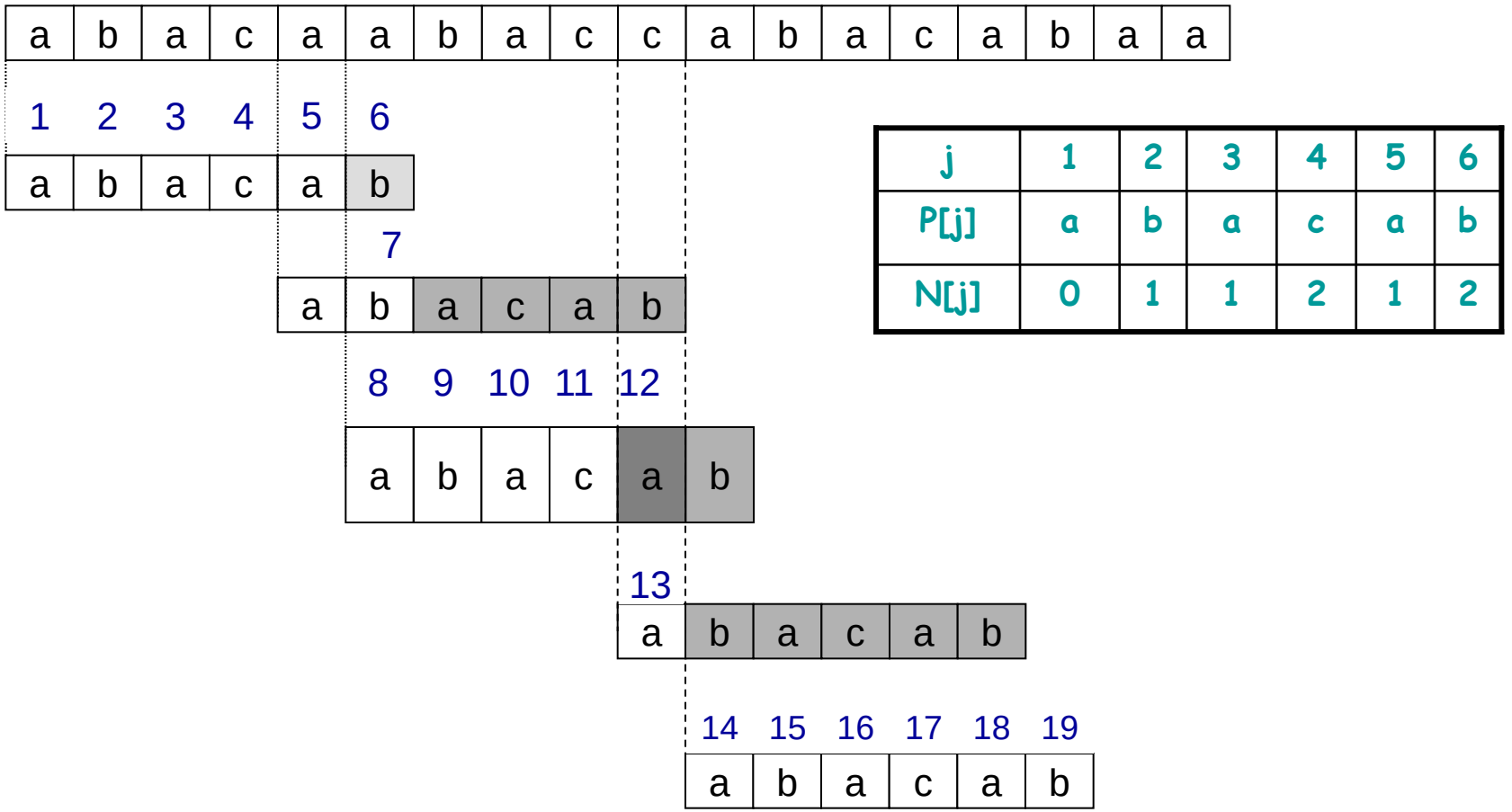
$$‘p_1p_2\cdots p_{k-1}’ = ’ p_{j-k+1}p_{j-k+2}\cdots p_{j-1}’ \qquad (4\text{-}4)$$

反之，若模式串中存在满足式（4-4）的两个子串，则匹配过程中产生‘失配’（即$s_i \neq p_j$）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，模式中头 k-1个字符的子串‘$p_1p_2\cdots p_{k-1}$’必定与主串中的第 i 个字符之前长度为 k-1的子串‘$s_{i-k+1}s_{i-k+2}\cdots s_{i-1}$’相等， 由此，匹配仅需从模式中第 k 个字符与主串中的第 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‘失配’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。由此可引出模式串的next函数的定义：

$$next[j] = \begin{cases} 0 & \text{当 } j = 1 \text{ 时} \\ \max\{k \mid 1 < k < j \text{ 且 } 'p_1 p_2 \cdots p_{k-1}' = 'p_{j-k+1} \cdots p_{j-1}'\} & \\ 1 & \text{其它情况} \end{cases}$$

KMP模式匹配：示例



j	1	2	3	4	5	6
P[j]	a	b	a	c	a	b
N[j]	0	1	1	2	1	2