3.2.2 蛮力字符串匹配

字符串匹配问题：

给定一个n个字符组成的串，称为**文本（text）**。
一个m个字符的串，称为**模式（pattern）**。

字符串匹配就是从**文本**中寻找匹配**模式**的**子串**。

更精确地说，我们找的是**文本**中的第一个**匹配子串的开始位置 *i***：使得$t_i=p_0,\ldots, t_{i+j}=p_j,\ldots, t_{i+m-1}=p_{m-1}$:

text T：

$t_0 \ldots t_i \ldots t_{i+j} \ldots t_{i+m-1} \ldots t_{n-1}$

pattern P：

$p_0 \ldots p_j \ldots p_{m-1}$

算法 **BruteForceStringMatch** (T[0..n-1], P[0..m-1])

//该算法实现了蛮力字符串匹配

//输入：一个n个字符的数组T[0..n-1]代表一段**文本**

//　　一个m个字符的数组P[0..n-1]代表一个**模式**

//输出：如果查找**成功**的话，返回文本的第一个匹配子串

//　　中第一个字符的位置；**否则**返回-1

```
for i←0 to n-m do
    j←0
    while j<m and P[j]=T[i+j] do
        j←j+1
        if  j=m  return i
return -1
```

蛮力字符串匹配的例子

N O B O D Y _ N O T I C E D _ H I M

N O T

N O T

N O T

N O T

N O T

N O T

N O T

N O T

蛮力字符串匹配算法 复杂度分析

```
算法  BruteForceStringMatch (T[0..n-1], P[0..m-1])
// ...
for  i←0 to n-m do
     j←0
     while  j<m  and  P[ j ] = T[ i+j ]  do
          j←j+1
          if  j=m   return i
return -1
```

最坏情况下，比较次数为：

$$C(n,m)=\sum_{i=0}^{n-m}\sum_{j=0}^{m}1=\sum_{i=0}^{n-m}m=nm-m^2$$

由于n>m，**最差复杂度**为：$\Theta(nm)$

在第7章中，我们将介绍 **Boyer-Moore 字符串匹配算法**，

其**最差复杂度**仅为：$\Theta(n+m)$

7.2 字符串匹配中的输入增强技术

- **字符串匹配**中的**输入增强**思想
  - **对模式进行预处理**，得到它的一些信息，然后在查找的过程中使用这些信息。

- **两种著名算法**
  - Knuth-Morris-Part（KMP)
  - Boyer-Moore（BM)
    - Horspool

7.2.1 Horspool算法

- 考虑在某些文本中查找模式BARBER

$s_0 s_1 \ldots$ C $\ldots$ s_{n-1}

BARBER

假设文本中对齐模式最后一个字符的元素是**字符C**

有四种情况：

$s_0 s_1 \ldots$ S $\ldots$ s_{n-1}

BARBER

BARBER

移动幅度等于模式长度

$s_0 s_1 \ldots$ B $\ldots$ s_{n-1}

BARBER

BARBER

模式中最右边的字符c和文本中的c对齐

$s_0 s_1 \ldots$ MER $\ldots$ s_{n-1}

LEADER

LEADER

移动幅度等于模式长度

$s_0 s_1 \ldots$ OR $\ldots$ s_{n-1}

REORDER

REORDER

把模式中前m-1个字符中最右边的c和文本中的c对齐

7.2.1 Horspool算法

- **模式的移动距离**

对于每一个字符c，移动距离t(c)为：

$$t(c)=\begin{cases}\text{模式的长度m，如果c不包含在模式的前m-1个字符中}\\ \text{模式前m-1个字符中最右边的c到模式最后一个字符的距离，其他}\end{cases}$$

```
算法 ShiftTable (P[0..m-1])
//为Horspool算法和 Boyer-Moore算法填充移动表
//输入：模式P[0..m-1]以及一个可能出现字符的字符表
//输出：以字母表中字符为索引的数组table[0..size-1]
for i←0 to size-1 do
    Table[i] ← m;
for j←0 to m-2 do
    Table[ P[j] ]←m-1-j;
return Table;
```

例如：

对于模式BARBER，

E,B,R,A 的移动距离分别为1,2,3,4，其他的为6

7.2.1 Horspool算法

- 算法步骤：

第一步：对于给定的长度为m的模式和在模式中用到的字母表，构造字符移动表 t；

第二步：将模式与文本的开头对齐；

第三步：从模式的最后一个字符开始，从后往前依次比较模式和文本对应的字符。如果所有的m个字符都能匹配，那么匹配成功，算法停止；

第四步：如果匹配不成功，找到和模式最后一个字符对齐的文本字符c，然后将模式沿着文本向右移动 t(c)个字符的距离。如果模式没有超出文本的最后一个字符，回到第三步，否则匹配失败，返回-1。

7.2.1 Horspool算法

```
算法 HorspoolMatching ( P[0..m-1], T[0...n-1] )
//实现 Horspool 字符串匹配算法
//输入：模式 P[0..m-1] 和文本 T[0...n-1]
//输出：匹配子串最左边字符的下标，没有匹配成功返回 -1
ShiftTable ( P[0..m-1] )
i ← m-1
while  i ≤ n-1 do
        k ← 0
        while k ≤ m-1 and  P[ m-1-k ]=T[ i-k ]  do
                k ← k+1
        if  k = m
                return  i - m+1
        else
                i ← i + Table[ T[i] ]
return  -1
```

7.2.1 Horspool算法

例：对于模式BARBER，

t(E) =1, t(B)=2, t(R)=3, t(A)=4；对其他字符X, t(X)=6

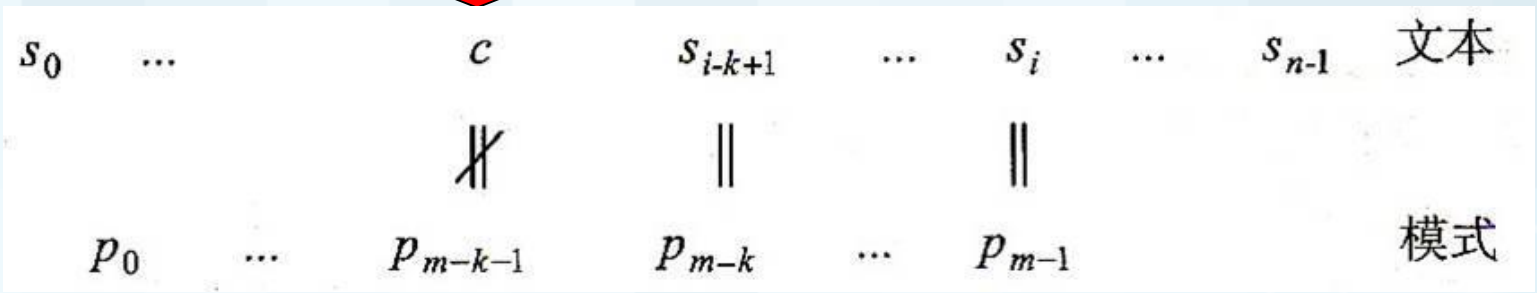
Horspool 运行结果:

```
JIM_SAW_ME_IN_A_BARBERSHOP
BARBER
     BARBER
       BARBER
              BARBER
                 BARBER
                    BARBER
JIM_SAW_ME_IN_A_BARBERSHOP
```

- Horspool 算法的最差效率Θ(mn)
- 但对于随机文本，它的效率为Θ(n)

7.2.2 Boyer-Moore算法

- 如果在遇到一个不匹配字符之前，如果已经有k(0<k<m)个字符匹配成功，则Boyer-Moore算法与Horspool算法处理结果不同。

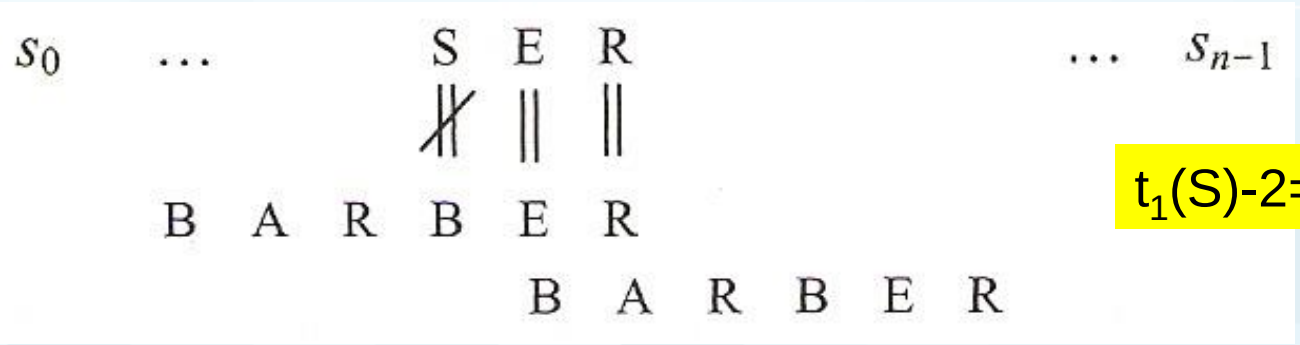


Boyer-Moore算法参考**两个数值**来确定移动距离。

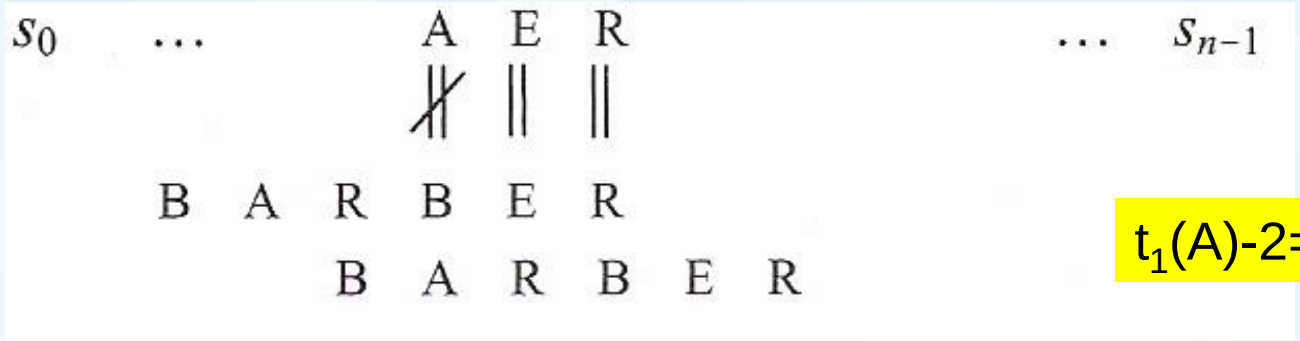
第一个数值是由文本中的不匹配字符c所确定，用公式**$t_1(c)-k$**来计算。称为**坏符号移动。**

其中**$t_1(c)$**是**Horspool算法**预先算好的值，k是**成功匹配**的字符个数

7.2.2 Boyer-Moore算法



$t_1(S)-2=6-2=4$



$t_1(A)-2=4-2=2$

$d_1=\max\{t_1(c)-k,1\}$

坏符号移动

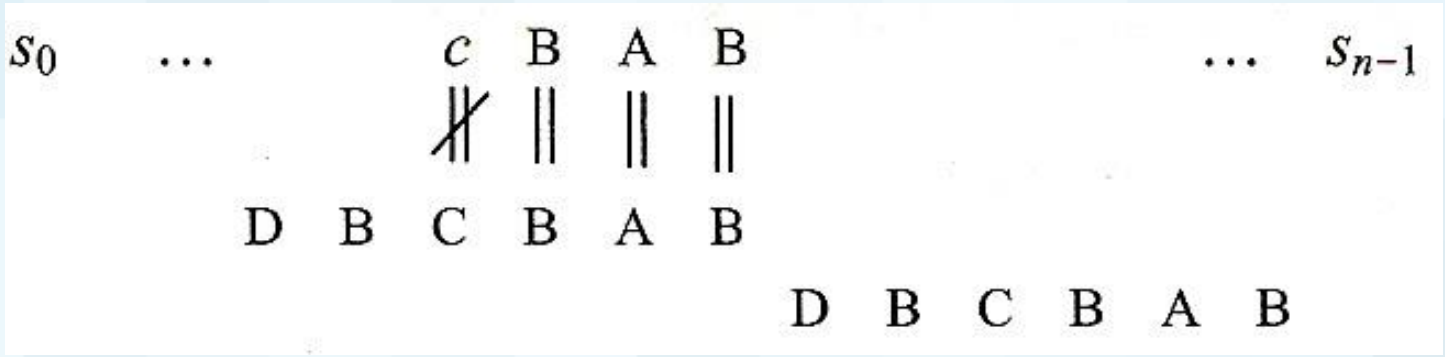
7.2.2 Boyer-Moore算法

- 第二个数值是由模式中最后k>0个成功匹配的字符所确定。称为好后缀移动。
- 把模式的结尾部分长度为k的好后缀记作 suff(k)
- 情况1：当模式中存在另外的suff(k)子串，且同最右边的子串相比，其前面一个字母不同。
- 移动距离 d_2 就是从这样的第二个最靠右的suff(k)子串到最右边的suff(k)子串的距离。

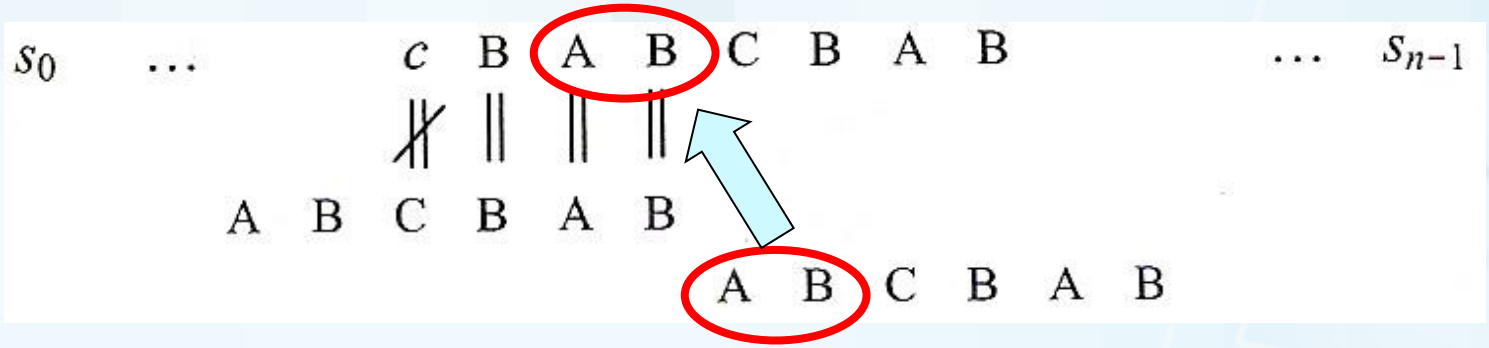
k	模式	d_2
1	ABCBAB	2
2	ABCDAB	4

7.2.2 Boyer-Moore算法

- 情况2：当模式中只存在1个suff(k)字串时：



k=3
移动6



问题：什么时候出现这种情况？

7.2.2 Boyer-Moore算法

- 为了避免情况2，我们需要找出长度为 $l<k$ 的最长前缀，它能够和长度同样为 l 的后缀完全匹配。
- 如果存在这样的前缀，我们通过求出这样的前缀和后缀之间的距离来作为移动距离d_2的值，否则移动距离就是m。

好后缀移动表

k	模式	d_2
1	ABCBAB	2
2	ABCBAB	4
3	ABCBAB	4
4	ABCBAB	4
5	ABCBAB	4

7.2.2 Boyer-Moore算法

第一步：对于给定的模式，构造坏符号移动表 t 和好后缀移动表d_2。

第二步：将模式与文本开始对齐

第三步：重复以下过程，比较模式和对应文本，直到匹配成功或者到达文本末尾。

（a）如果从模式末端开始只有k<m个字符匹配，则模式移动距离为d：

$$d = \begin{cases} d_1 & if \quad k = 0 \\ \max\{d_1, d_2\} & if \quad k > 0 \end{cases}$$

$$where \quad d_1 = \max\{t_1(c) - k, 1\}$$

（b）如果所有m个字符都匹配，返回成功匹配的位置；

7.2.2 Boyer-Moore算法：举例

- 在一个由英文字母和空格构成的文本中查找BAOBAB

坏符号移动表

c	A	B	C	D	...	O	...	Z	-
$t_1(c)$	1	2	6	6	6	3	6	6	6

好后缀移动表

k	模式	d_2
1	BAOB$\bar{A}$B	2
2	$\bar{B}$AOBAB	5
3	$\bar{B}$AOBAB	5
4	$\bar{B}$A$\bar{O}$BAB	5
5	$\bar{B}$AOBAB	5

B E S S _ K N E W _ A B O U T _ B A O B A B S

B A O B A B

$d_1 = t_1(K) - 0 = 6$

B A O B A B

$d_1 = t_1(\_) - 2 = 4$

$d_2 = 5$

$d = \max\{4, 5\} = 5$

B A O B A B

$d_1 = t_1(\_) - 1 = 5$

$d_2 = 2$

$d = \max\{5, 2\} = 5$

B A O B A B

课堂练习

- 对于Boyer-Moore算法，设 Σ={0，1}。
- 计算模式 **011011011** 所对应的
  - 坏符号移动表
  - 好后缀移动表