

## 4.3 串的模式匹配



- **成功**是指在目标串s中找到一个模式串t——t是s的子串，返回t在s中的位置。
- **不成功**则指目标串s中不存在模式串t——t不是s的子串，返回-1。

## 4.4.1 Brute-Force算法

Brute-Force简称为**BF算法**，亦称简单匹配算法。采用穷举的思路。

s: 

a	a	a	a	b	c	d
---	---	---	---	---	---	---

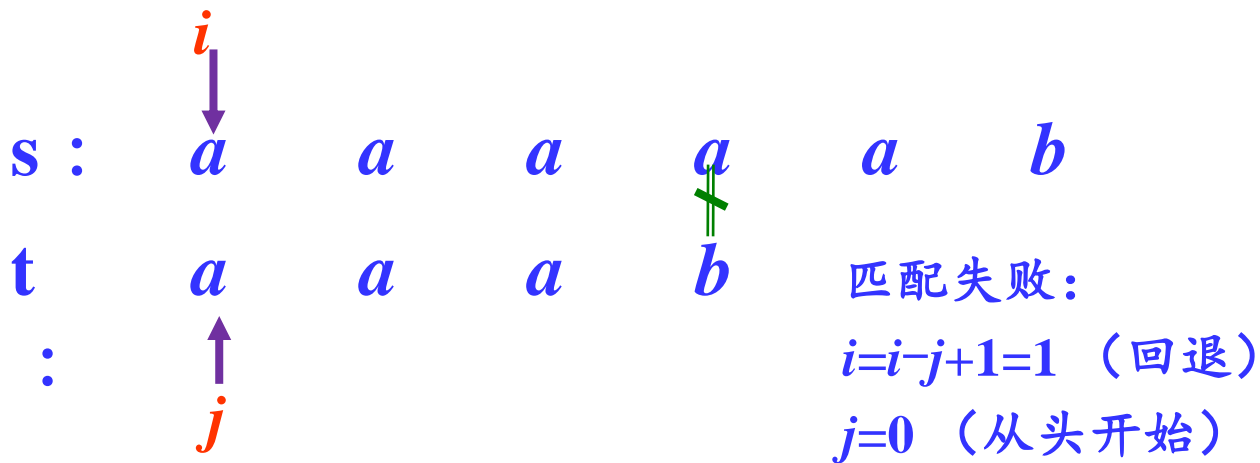
t: 

a	a	a	b	c	<del>c</del>
---	---	---	---	---	--------------

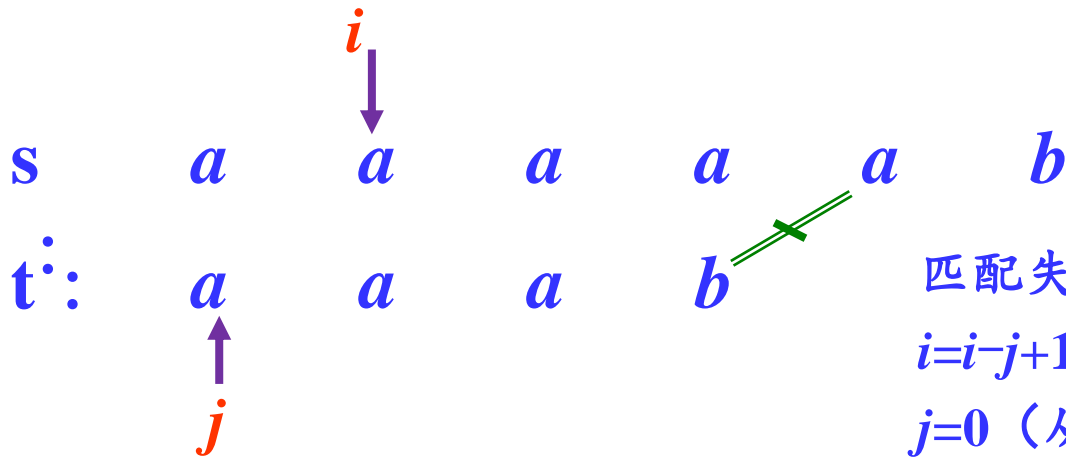
 ✓

匹配成功

例如，设目标串 $s$ ="aaaaab"，模式串 $t$ ="aaab"。 $s$ 的长度为 $n$  ( $n=6$ )， $t$ 的长度为 $m$  ( $m=4$ )。BF算法的匹配过程如下。



$i=1, j=0$



匹配失败:

$i=i-j+1=2$  (回退)

$j=0$  (从头开始)

$i=2, j=0$

s	a	a	a	a	a	b
t:	a	a	a	b		

$j$  ↑       $i$  ↓

匹配成功:

$i=6, j=4$

返回  $i-t.length=2$

对应的BF算法如下：

```
int index(SqString s,SqString t)
{   int i=0, j=0;
    while (i<s.length && j<t.length)
    {   if (s.data[i]==t.data[j])           //继续匹配下一个字符
        {   i++;                          //主串和子串依次匹配下一个字符
            j++;
        }
        else                               //主串、子串指针回溯重新开始下一次匹配
        {   i=i-j+1;                       //主串从下一个位置开始匹配
            j=0;                           //子串从头开始匹配
        }
    }
    if (j>=t.length)
        return(i-t.length);               //返回匹配的第一个字符的下标
    else
        return(-1);                       //模式匹配不成功
}
```

## BF算法分析：

- 算法在字符比较不相等，需要回溯（即 $i=i-j+1$ ）：即退到s中的下一个字符开始进行继续匹配。
- 最好情况下的时间复杂度为 $O(m)$ 。
- 最坏情况下的时间复杂度为 $O(n \times m)$ 。
- 平均的时间复杂度为 $O(n \times m)$ 。

## 4.3.2 KMP算法

KMP算法是D.E.Knuth、J.H.Morris和V.R.Pratt共同提出的，简称**KMP算法**。

该算法较BF算法有较大改进，主要是**消除了主串指针的回溯**，从而使算法效率有了某种程度的提高。

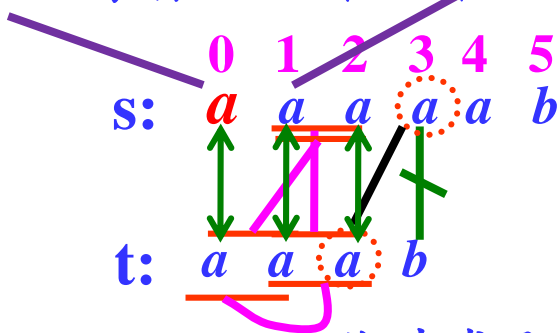


## KMP算法用next数组保存部分匹配信息的演示

目标串s=“aaaaab”，模式串t=“aaab”。

## 开始匹配的字符

## 下次开始匹配的字符



从t中发现: *b*前面有2个字符和开头的2个字符相同



用一个数组next保存：next[3]=2



下次匹配的字符: `s[3]`和`t[next[3]]`即`t[2]`

$\text{next}[j]$ 是指 $t[j]$ 字符前有多少个字符与 $t$ 开头的字符相同。

模式串 $t$ 存在某个 $k$  ( $0 < k < j$ )，使得以下成立：

$$\underbrace{\text{"}t_0t_1\dots t_{k-1}\text{"}}_{\text{开头的}k\text{个字符}} = \underbrace{\text{"}t_{j-k}t_{j-k+1}\dots t_{j-1}\text{"}}_{t[j]\text{前面的}k\text{个字符}}$$

例如， $t = \overset{0}{a} \overset{1}{b} \overset{2}{a} \overset{3}{b} \overset{4}{c}$  考虑 $t[4]='c'$



有 $t_0t_1 = t_2t_3 = "ab"$   $\Rightarrow k=2$

所以 $\text{next}[4] = k = 2$ 。

归纳起来，定义next[j]数组如下：

$$\text{next}[j] = \begin{cases} \text{MAX}\{k \mid 0 < k < j, \text{ 且 } \underbrace{t_0 t_1 \dots t_{k-1}}_{\text{开头的}k\text{个字符}} = \underbrace{t_{j-k} t_{j-k+1} \dots t_{j-1}}_{\text{后面的}k\text{个字符}}\} & \text{当此集合非空时} \\ -1 & \text{当} j=0 \text{ 时} \\ 0 & \text{其他情况} \end{cases}$$

t = "aaab" 对应的next数组如下：

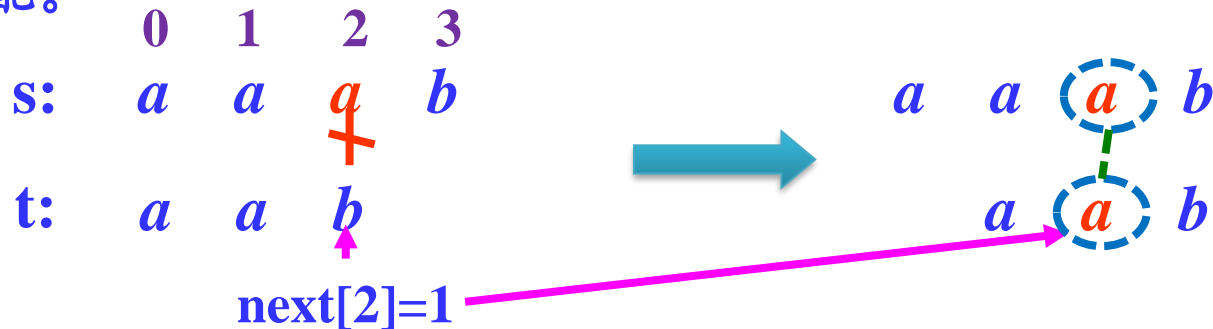
j	0	1	2	3
t[j]	a	a	a	b
next[j]	-1	0	1	2

$t_0 = t_1 = "a"$   $t_0 t_1 = t_1 t_2 = "aa"$

## next[j]的含义

### (1) $\text{next}[j]=k$ 表示什么信息?

说明模式串 $t[j]$ 之前有 $k$ 个字符已成功匹配，下一趟应从 $t[k]$ 开始匹配。



### (2) $\text{next}[j]=-1$ 表示什么信息?

说明模式串 $t[j]$ 之前没有任何用于加速匹配的信息，下一趟应从 $t$ 的开头即 $j++ \Rightarrow j=0$ 开始匹配。

如 $t = \text{"abcd"}$ ， $\text{next}[0]=\text{next}[1]=\text{next}[2]=\text{next}[3]=-1$ 。

由模式串t求next值的算法：

```
void GetNext(SqString t,int next[])
{   int j, k;
    j=0; k=-1; next[0]=-1;
    while (j<t.length-1)
    {   if (k==-1 || t.data[j]==t.data[k])
        {   j++; k++;
            next[j]=k;
        }
        else k=next[k];
    }
}
```

## KMP算法:

```
int KMPIIndex(SqString s,SqString t)
```

```
{  int next[MaxSize], i=0, j=0;
```

```
    GetNext(t,next);
```

```
    while (i<s.length && j<t.length)
```

```
    {
```

```
        if (j==-1 || s.data[i]==t.data[j])
```

```
        {  i++;
```

```
            j++;
```

//i、j各增1

```
        }
```

```
        else j=next[j];
```

//i不变, j后退

```
    }
```

没有有用信息或两个字符相等时, 继续比较后面的字符

```
    return(i-t.length);
```

//返回匹配模式串的首字符下标

```
    else
```

```
        return(-1);
```

//返回不匹配标志

```
}
```

主串位置不变, 子串重新定位 (右移)

## KMP算法分析

设串 $s$ 的长度为 $n$ ，串 $t$ 长度为 $m$ 。

在KMP算法中求next数组的时间复杂度为 $O(m)$ ，在后面的匹配中因主串 $s$ 的下标不减即不回溯，比较次数可记为 $n$ ，所以KMP算法平均时间复杂度为 $O(n+m)$ 。

最坏的时间复杂度为 $O(n \times m)$ 。

【例4-3】 已知字符串S为“*abaabaabacacaabaabcc*”，模式串t为“*abaabc*”，采用KMP算法进行匹配，第一次出现“失配” ( $s[i] \neq t[j]$ )时， $i=j=5$ ，则下次开始匹配时， $i$ 和 $j$ 的值分别是\_\_\_\_\_。

A. $i=1, j=0$     B. $i=5, j=0$     C. $i=5, j=2$     D. $i=6, j=2$

说明：本题为2015年全国考研题

$j$	0	1	2	3	4	5
$t[j]$	<i>a</i>	<i>b</i>	<i>a</i>	<i>a</i>	<i>b</i>	<i>c</i>
$next[j]$	-1	0	0	1	1	2

选C



## 思考题

上述KMP算法仍然存在什么缺陷？

设目标串s=“*aaabaaaab*”，模式串t=“*aaaab*”。KMP模式匹配过程。

求t的next:

j	0	1	2	3	4
t[j]	<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>b</i>
next[j]	-1	0	1	2	3

j	0	1	2	3	4
t[j]	<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>b</i>
next[j]	-1	0	1	2	3

s:    0 1 2 3 4 5 6 7 8  
       *a a a b a a a a b*  
       ↑ ↑ ↑  
 t:    *a a a a b*  
       0 1 2 3 4

失败:  
*i=3*  
*j=3, j=next[3]=2*

j	0	1	2	3	4
t[j]	<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>b</i>
next[j]	-1	0	1	2	3

*i*=3  
*j*=2

	0	1	2	3	4	5	6	7	8
<i>s</i> :	<i>a</i>	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>b</i>
<i>t</i> :	<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>b</i>				
	0	1	2	3	4				

A pink 'X' is drawn over the character 'a' at index 3 of string *s* and the character 'a' at index 3 of string *t*.

失败:  
*i*=3  
*j*=2, *j*=next[2]=1

j	0	1	2	3	4
t[j]	<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>b</i>
next[j]	-1	0	1	2	3

$i=3$   
 $j=1$

	0	1	2	3	4	5	6	7	8
$s:$	<i>a</i>	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>b</i>
$t:$		<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>b</i>			
	0	1	2	3	4				

✗


失败:  
 $i=3$   
 $j=1, j=\text{next}[1]=0$

j	0	1	2	3	4
t[j]	<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>b</i>
next[j]	-1	0	1	2	3

*i*=3  
*j*=0

*s*:    0 1 2 3 4 5 6 7 8  
       *a a a b a a a a b*

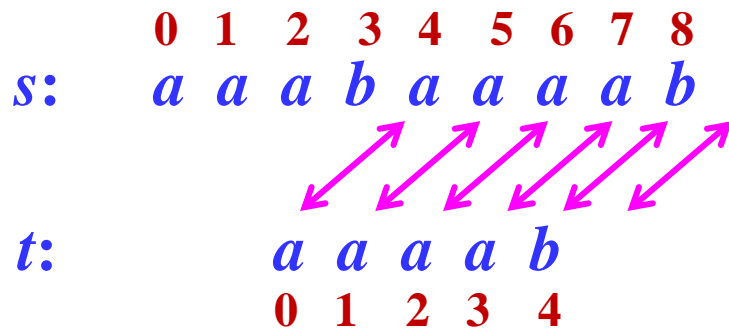
*t*:            *a a a a b*  
               0 1 2 3 4



失败:  
*i*=3  
*j*=0, *j*=next[0]=-1

j	0	1	2	3	4
t[j]	<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>b</i>
next[j]	-1	0	1	2	3

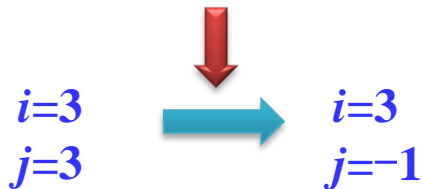
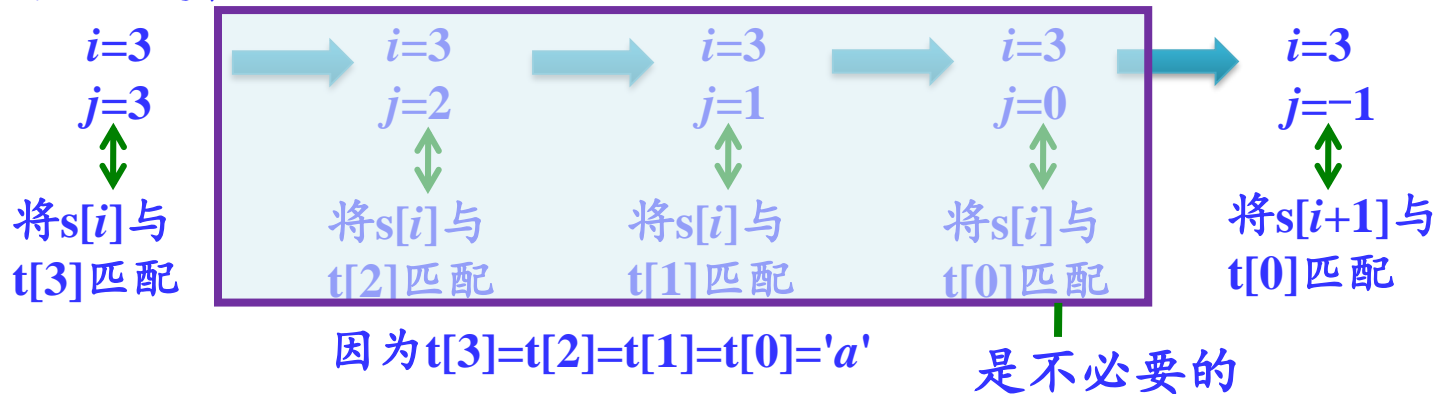
因为  $j = -1$ :  
 $i++$ ;  
 $j++$ ;



成功:  
返回4

j	0	1	2	3	4
t[j]	<i>a</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>b</i>
next[j]	-1	0	1	2	3

前面的匹配过程：





将next改为nextval:

j	0	1	2	3	4
t[j]	a	a	a	a	b
next[j]	-1	0	1	2	3
nextval[j]	-1	-1	-1	-1	3

$\text{next}[1]=0$

$t[1]=t[\text{next}[1]]=t[0]='a'$

$\therefore \text{nextval}[1]=\text{nextval}[0]=-1$

$t[4]='b' \neq t[\text{next}[4]]=t[3]='a'$

$\therefore \text{nextval}[4]=\text{next}[4]$



- $\text{nextval}[0]=-1$
- 当  $t[j]=t[\text{next}[j]]$  时:  $\text{nextval}[j]=\text{nextval}[\text{next}[j]]$
- 否则:  $\text{nextval}[j]=\text{next}[j]$

用nextval取代next, 得到改进的KMP算法。

## 使用改进后的KMP算法示例：

j	0	1	2	3	4
t[j]	a	a	a	a	b
nextval[j]	-1	-1	-1	-1	3

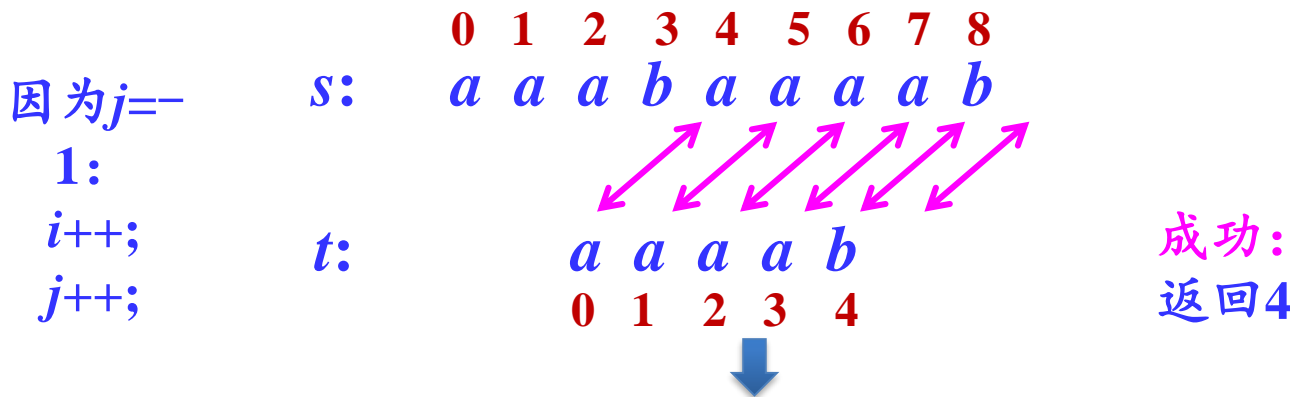
	0	1	2	3	4	5	6	7	8
s:	a	a	a	b	a	a	a	a	b
	↕	↕	↕						
t:	a	a	a	a	b				
	0	1	2	3	4				

失败：

$i=3$

$j=3, j=\text{nextval}[3]=-1$

j	0	1	2	3	4
t[j]	a	a	a	a	b
nextval[j]	-1	-1	-1	-1	3



改进后的KMP算法进一步提高模式匹配的效率。

# 数据结构经典算法的启示

BF算法



利用模式串中部分匹配信息

KMP算法



——本章完——