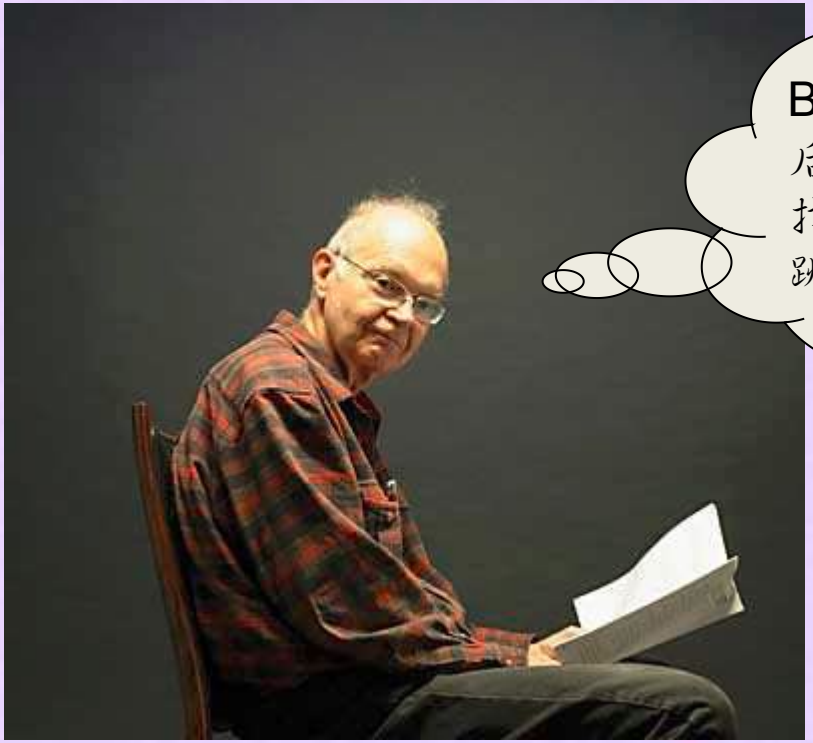
本节主题：

串的模式匹配(KMP算法)

KMP算法，经典模式匹配算法



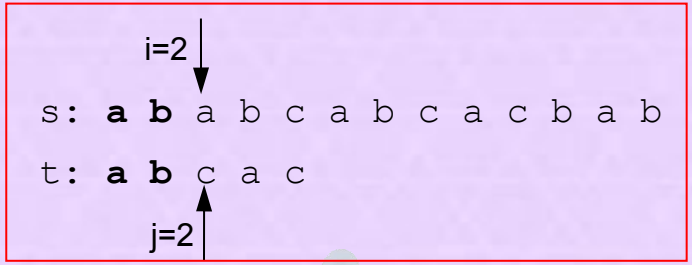
B-F算法中，匹配失败后不必完全从头再来，找到可以利用的信息，跳跃性匹配……

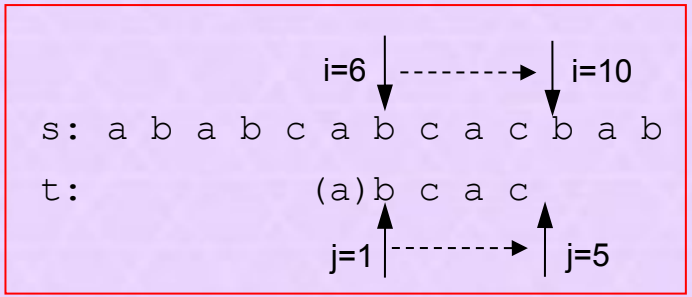
D.E.Knuth

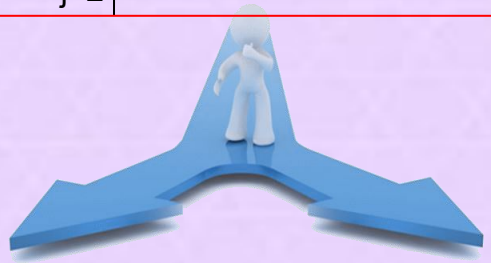
J.H.Morris

V.R.Pratt

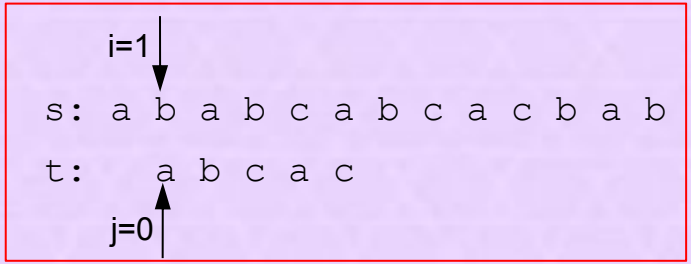
发现B-F算法中的改进点

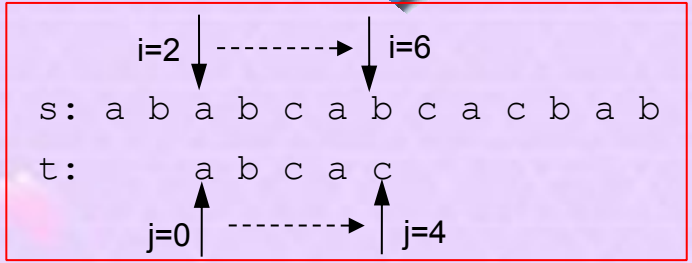






i不再回退,j从哪里开始?





i不再回退，j从哪里开始？

- 设置模式串的“部分匹配”信息！
  - 以t="abcac"为例
  - 令next[j]=k，表示当模式t中第j个字符与主串s中的相应字符“失配”时，在模式t中需重新和主串s中该字符进行比较的字符的位置

j	0	1	2	3	4
t.dada[*j*]	*a*	*b*	*c*	*a*	*c*
next[*j*]	-1	0	0	0	1

c前字符b不等于t的开头字符a

前面的子串bc、c都不与t的开头字符a匹配

前面的子串bca、ca和a，只有a与t的开头字符a匹配

$$next[j]=\begin{cases} -1 & \text{当j=0时} \\ Max\{k \mid 0<k<j \text{且} "t_0t_1\ldots t_{k-1}"="t_{j-k}t_{j-k-1}\ldots t_{j-1}"\} & \text{当此集合非空} \\ 0 & \text{其他情况} \end{cases}$$

理解next[j]

$$next[j]=\begin{cases} -1 & \text{当j=0时} \\ Max\{\ k \mid 0<k<j \text{且} "t_0t_1...t_{k-1}"="t_{j-k}t_{j-k-1}...t_{j-1}\} & \text{当此集合非空} \\ 0 & \text{其他情况} \end{cases}$$

- t=abaabcac

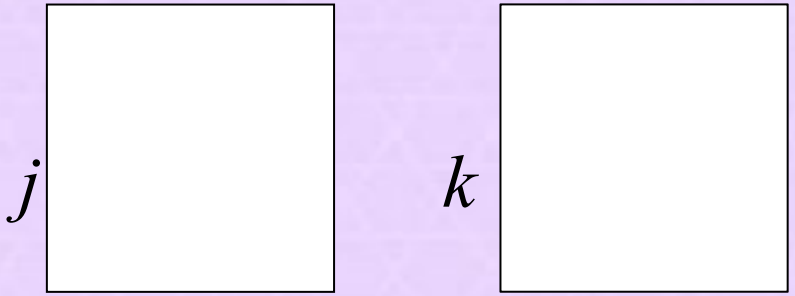
j	0	1	2	3	4	5	6	7
t.data[*j*]	*a*	*b*	*a*	*a*	*b*	*c*	*a*	*c*
next[*j*]	-1	0	0	1	1	2	0	1

- t=aaab

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

算法：由模式串 t 求出next值

```
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];
    }
}
```



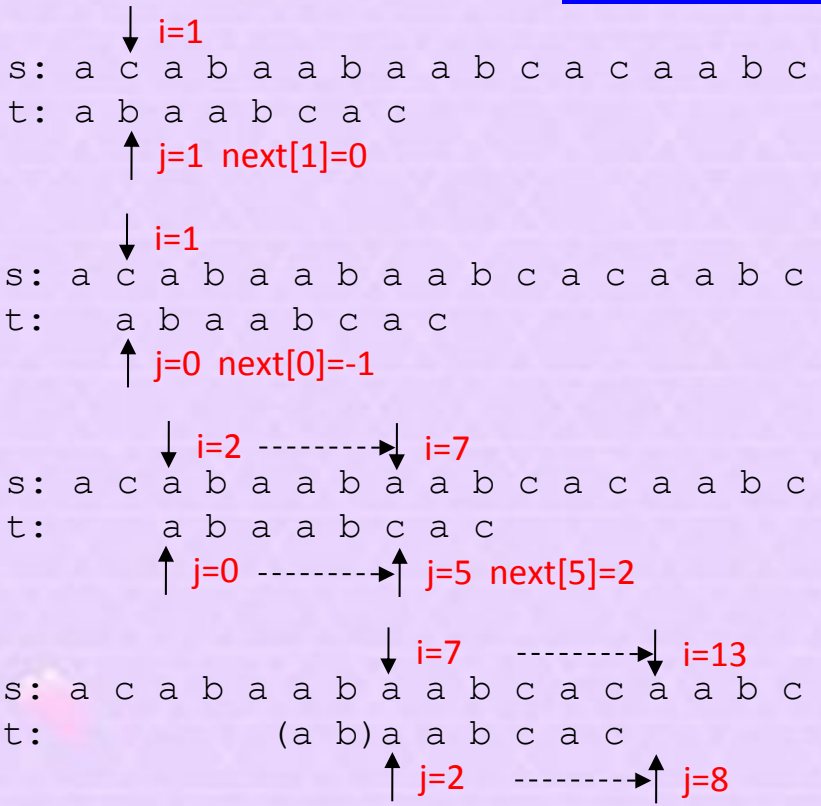
j	0	1	2	3	4	5	6	7
t.data[*j*]	*a*	*b*	*a*	*a*	*b*	*c*	*a*	*c*
next[*j*]	-1	0	0	1	1	2	0	1

KMP算法

```
int KMPIndex(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++;
      }
      else
         j=next[j];
   }
   if (j>=t.length)
      return(i-t.length);
   else
      return(-1);
}
```

j	0	1	2	3	4	5	6	7
t.data[j]	a	b	a	a	b	c	a	c
next[j]	-1	0	0	1	1	2	0	1

主串s的长度为n，
子串t长度为m，
KMP算法总的时间复杂度为O(n+m)



KMP算法运行再例

j	0	1	2	3	4
t.data[j]	*a*	*a*	*a*	*a*	*b*
next[j]	-1	0	1	2	3

- 待改进的问题
  - 模式“aaaab”在和主串“aaabaaaab”匹配时，当i=3,j=3时，s.data[3]≠t.data[3]，由next[j]的指示还需进行i=3、j=2,i=3、j=1,i=3、j=0等3次比较。
- 解决思路
  - 模式中的第4个字符与前3个字符相同，不需要再和主串中第4个字符相比较，而是直接进行i=4，j=0时的字符比较。

```
s:aaabaaaab  i=3
t:aaaab      j=3, next[3]=2
```

```
s:aaabaaaab  i=3
t: aaaab     j=2, next[2]=1
```

```
s:aaabaaaab  i=3
t:  aaaab    j=1, next[1]=0
```

```
s:aaabaaaab  i=3
t:   aaaab   j=0, next[0]= -1
```

```
s:aaabaaaab  i=4  ------>  i=9
t:    aaaab  j=0          j=5
```

使用修正的nextval

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

- 思路
  - 若按上述定义得到next[j]=k，而模式中$t_j=t_k$，则当匹配失败时，不需要再和t_k进行比较，而直接和$t_{next[k]}$进行比较。即，此时的next[j]应和next[k]相同
- 解决
  - 比较t.data[j]和t.data[k]
  - 若不等，则 nextval[j]=k；
  - 若相等nextval[j]=nextval[k]。

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

修改后的KMP算法

```
int KMPIndex1(SqString s,SqString t)
{
    int nextval[MaxSize],i=0,j=0;
    GetNextval(t,nextval);
    while (i<s.length && j<t.length)
    {
        if (j==-1 || s.data[i]==t.data[j])
        {
            i++;
            j++;
        }
        else
            j=nextval[j];
    }
    if (j>=t.length)
        return(i-t.length);
    else
        return(-1);
}
```

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

```
s:aaabaaaab
t:aaaab
```
i=3

j=3, next[3]=-1

```
s:aaabaaaab
t:    aaaab
```
i=4 ⇢ i=9

j=0 j=5

思考题

- KMP算法给我们什么启示？