本节内容

KMP算法

求next数组

1

KMP算法代码

```
int Index_KMP(SString S,SString T,int next[]){
    int i=1, j=1;
    while(i<=S.length&&j<=T.length){
        if(j==0||S.ch[i]==T.ch[j]){
            ++i;
            ++j;                    //继续比较后继字符
        }
        else
            j=next[j];              //模式串向右移动
    }
    if(j>T.length)
        return i-T.length;          //匹配成功
    else
        return 0;
}
```

g	o	o	g	l	e
1	2	3	4	5	6

j

由模式串确定的next数组

int next[7];

0	1	2	3	4	5	6
	0	1	1	1	2	1

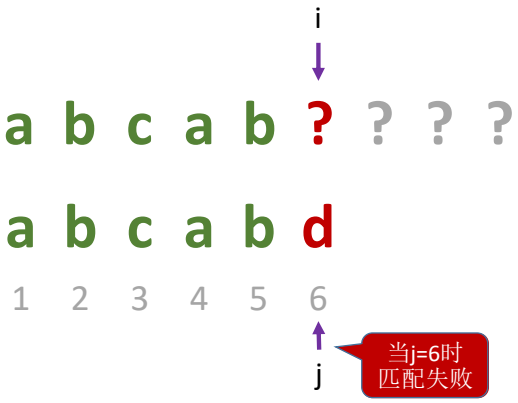
当 j=k 且发现字符不匹配时，令j=next[k]

2

观察：求模式串的next数组

next 数组：当模式串的第 j 个字符匹配失败时，令模式串跳到 next[j] 再继续匹配

模式串：’abcabd’

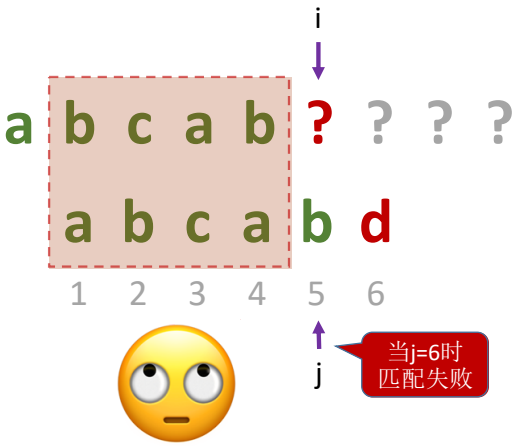


3

观察：求模式串的next数组

next 数组：当模式串的第 j 个字符匹配失败时，令模式串跳到 next[j] 再继续匹配

模式串：’abcabd’

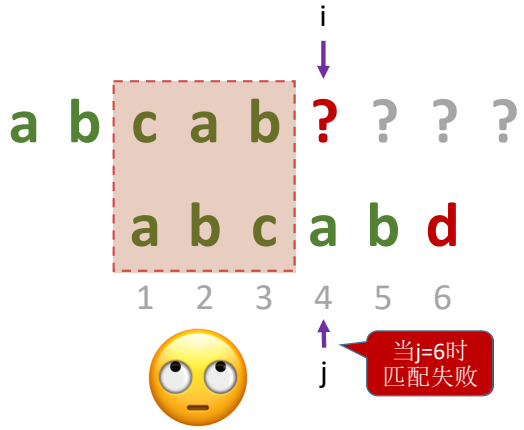


4

观察：求模式串的next数组

next 数组：当模式串的第 j 个字符匹配失败时，令模式串跳到 next[j] 再继续匹配

模式串：’abcabd’

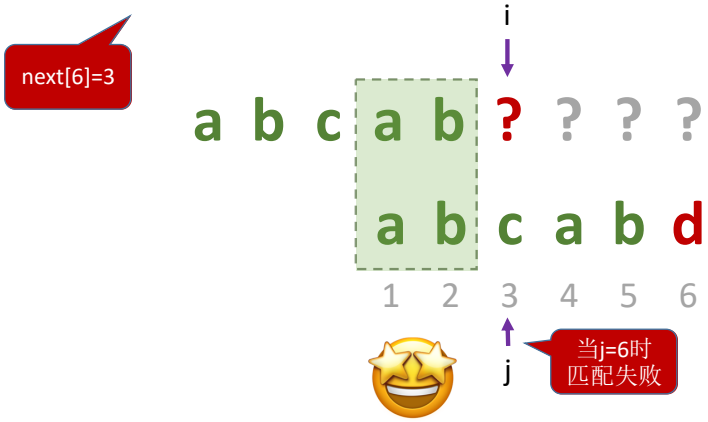


5

观察：求模式串的next数组

next 数组：当模式串的第 j 个字符匹配失败时，令模式串跳到 next[j] 再继续匹配

模式串：’abcabd’

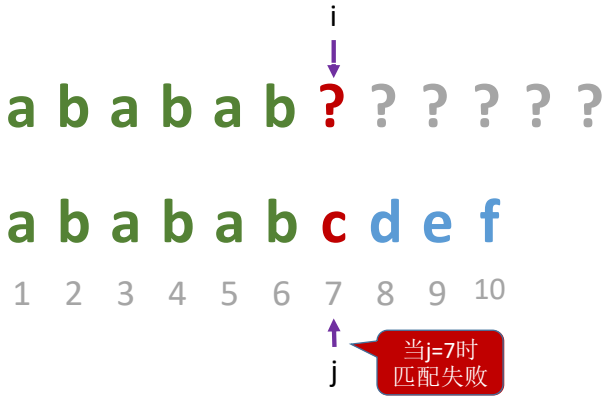


6

观察：求模式串的next数组

next 数组：当模式串的第 j 个字符匹配失败时，令模式串跳到 next[j] 再继续匹配

模式串：’abababcdef’

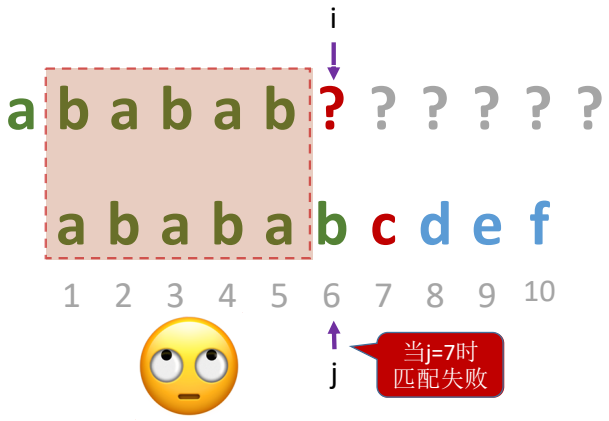


7

观察：求模式串的next数组

next 数组：当模式串的第 j 个字符匹配失败时，令模式串跳到 next[j] 再继续匹配

模式串：’abababcdef’

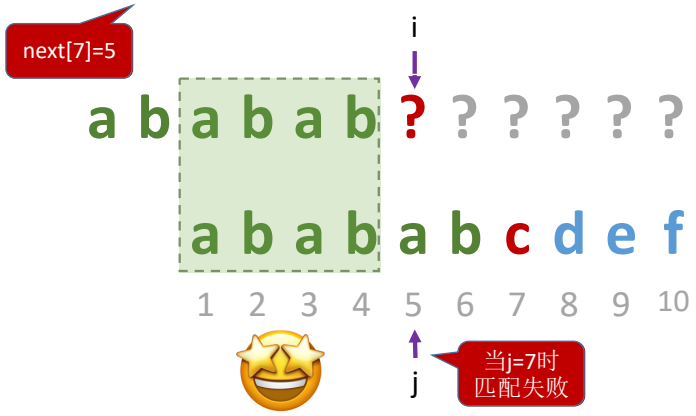


8

观察：求模式串的next数组

next 数组：当模式串的第 j 个字符匹配失败时，令模式串跳到 next[j] 再继续匹配

模式串：’abababcdef’

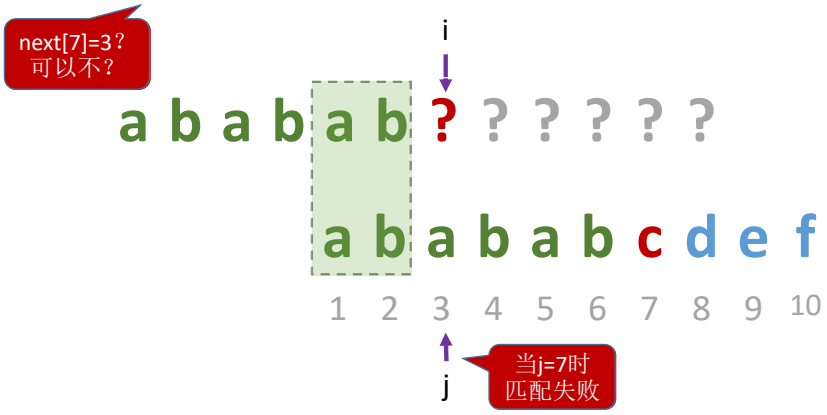


9

观察：求模式串的next数组

next 数组：当模式串的第 j 个字符匹配失败时，令模式串跳到 next[j] 再继续匹配

模式串：’abababcdef’

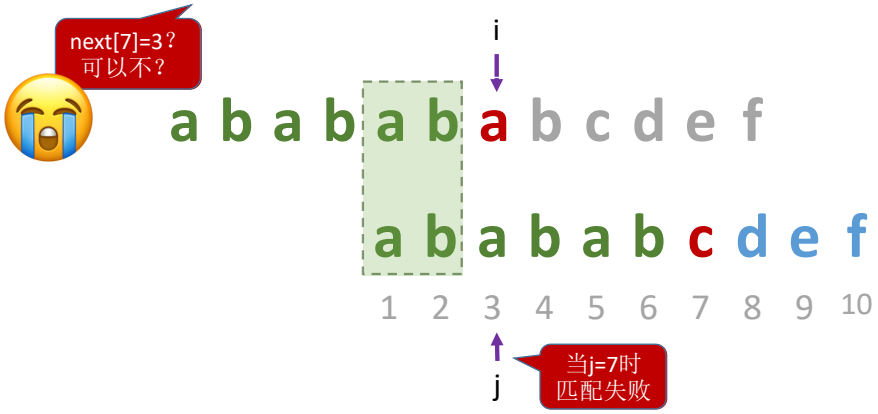


10

观察：求模式串的next数组

next 数组：当模式串的第 j 个字符匹配失败时，令模式串跳到 next[j] 再继续匹配

模式串：’abababcdef’

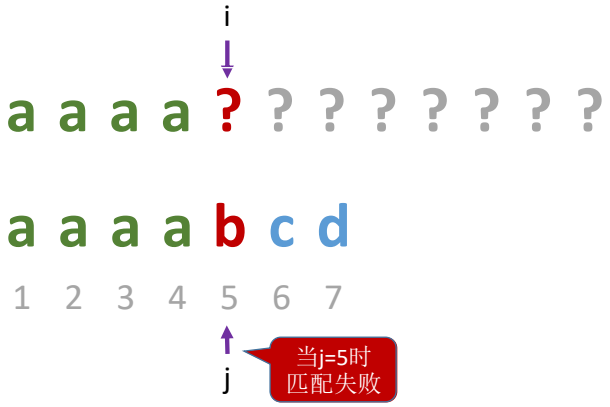


11

观察：求模式串的next数组

next 数组：当模式串的第 j 个字符匹配失败时，令模式串跳到 next[j] 再继续匹配

模式串：’aaaabcd’

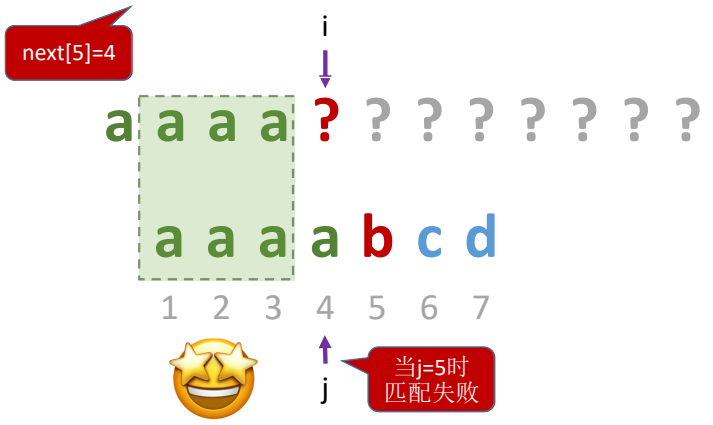


12

观察：求模式串的next数组

next 数组：当模式串的第 j 个字符匹配失败时，令模式串跳到 next[j] 再继续匹配

模式串：’aaaabcd’

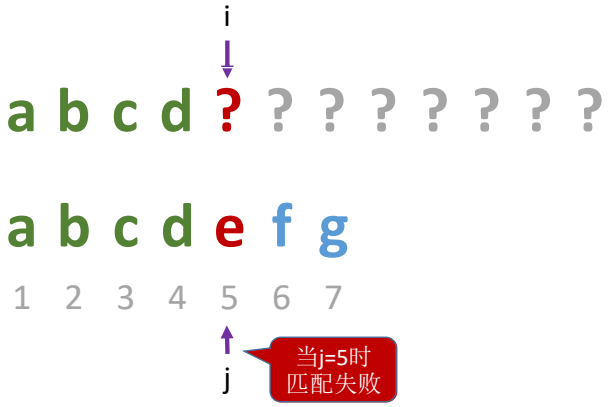


13

观察：求模式串的next数组

next 数组：当模式串的第 j 个字符匹配失败时，令模式串跳到 next[j] 再继续匹配

模式串：’abcdefg’

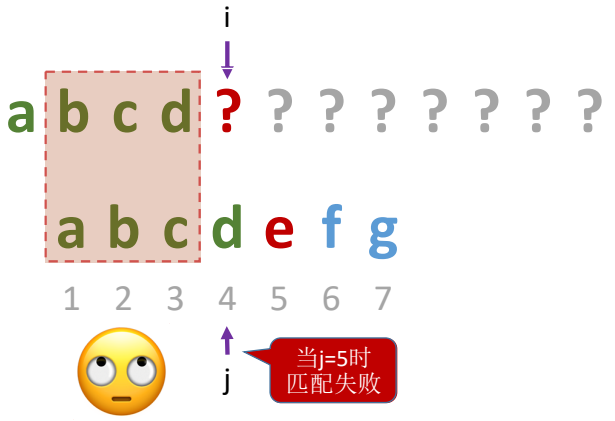


14

观察：求模式串的next数组

next 数组：当模式串的第 j 个字符匹配失败时，令模式串跳到 next[j] 再继续匹配

模式串：’abcdefg’

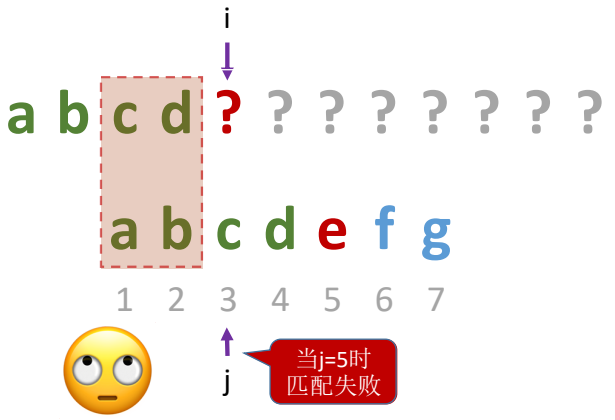


15

观察：求模式串的next数组

next 数组：当模式串的第 j 个字符匹配失败时，令模式串跳到 next[j] 再继续匹配

模式串：’abcdefg’

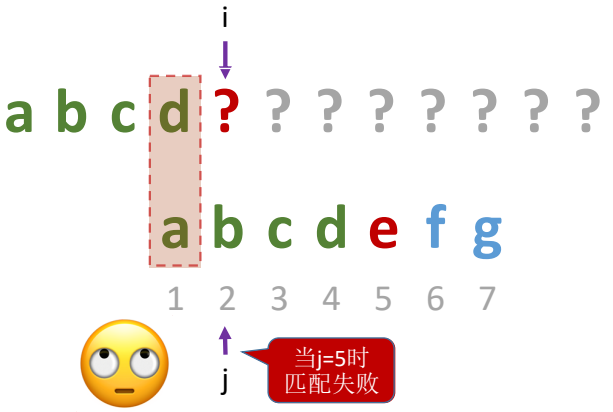


16

观察：求模式串的next数组

next 数组：当模式串的第 j 个字符匹配失败时，令模式串跳到 next[j] 再继续匹配

模式串：’abcdefg’

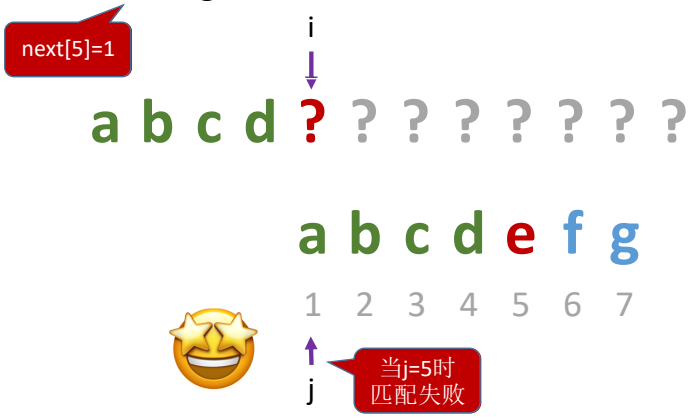


17

观察：求模式串的next数组

next 数组：当模式串的第 j 个字符匹配失败时，令模式串跳到 next[j] 再继续匹配

模式串：’abcdefg’

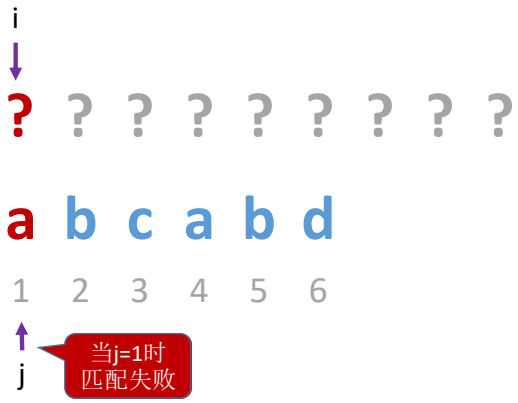


18

观察：求模式串的next数组

next 数组：当模式串的第 j 个字符匹配失败时，令模式串跳到 next[j] 再继续匹配

模式串：’abcabd’

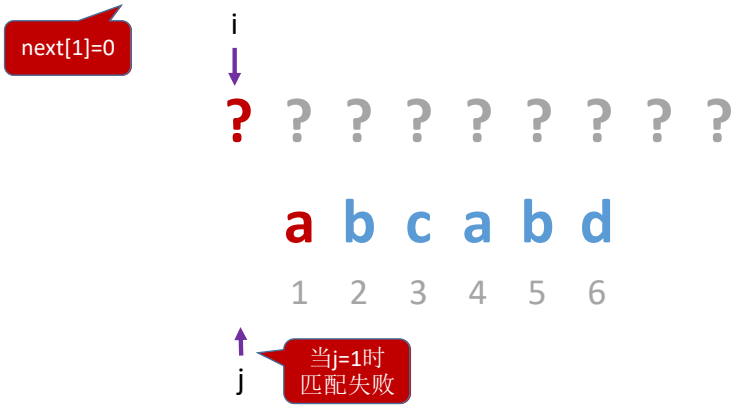


19

观察：求模式串的next数组

next 数组：当模式串的第 j 个字符匹配失败时，令模式串跳到 next[j] 再继续匹配

模式串：’abcabd’



20

KMP算法代码

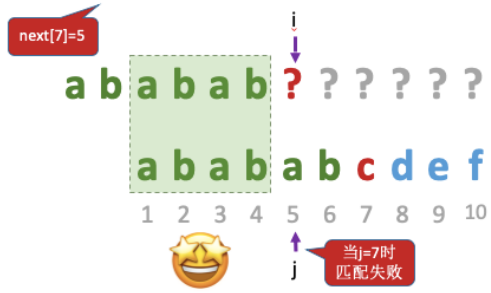
```
int Index_KMP(SString S,SString T,int next[]){
    int i=1, j=1;
    while(i<=S.length&&j<=T.length){
        if(j==0||S.ch[i]==T.ch[j]){
            ++i;
            ++j;                    //继续比较后继字符
        }
        else
            j=next[j];              //模式串向右移动
    }
    if(j>T.length)
        return i-T.length;          //匹配成功
    else
        return 0;
}
```


21

总结：求模式串的next数组

next 数组：当模式串的第 j 个字符匹配失败时，令模式串跳到 next[j] 再继续匹配

模式串：'abababcdef'



模式串：'abcdefg'

串的前缀：包含第一个字符，且不包含最后一个字符的子串
串的后缀：包含最后一个字符，且不包含第一个字符的子串

当第j个字符匹配失败，由前 1~j-1 个字符组成的串记为S，则：
next[j]= S的最长相等前后缀长度+1
特别地， next[1]=0

22

练习：求模式串的next数组

串的前缀：包含第一个字符，且不包含最后一个字符的子串
串的后缀：包含最后一个字符，且不包含第一个字符的子串
当第j个字符匹配失败，由前 1~j-1 个字符组成的串记为S，则：
next[j]= S的最长相等前后缀长度+1
特别地，next[1]=0

模式串：’ababaa’

序号j	1	2	3	4	5	6
模式串	a	b	a	b	a	a
next[j]						

23

练习：求模式串的next数组

串的前缀：包含第一个字符，且不包含最后一个字符的子串
串的后缀：包含最后一个字符，且不包含第一个字符的子串
当第j个字符匹配失败，由前 1~j-1 个字符组成的串记为S，则：
next[j]= S的最长相等前后缀长度+1
特别地，next[1]=0

模式串：’ababaa’

序号j	1	2	3	4	5	6
模式串	a	b	a	b	a	a
next[j]	0	1	1	2	3	4

24

练习：求模式串的next数组

串的前缀：包含第一个字符，且不包含最后一个字符的子串
串的后缀：包含最后一个字符，且不包含第一个字符的子串
当第j个字符匹配失败，由前 1~j-1 个字符组成的串记为S，则：
next[j]= S的最长相等前后缀长度+1
特别地，next[1]=0

模式串：’aaaab’

序号j	1	2	3	4	5
模式串	a	a	a	a	b
next[j]					

25

练习：求模式串的next数组

串的前缀：包含第一个字符，且不包含最后一个字符的子串
串的后缀：包含最后一个字符，且不包含第一个字符的子串
当第j个字符匹配失败，由前 1~j-1 个字符组成的串记为S，则：
next[j]= S的最长相等前后缀长度+1
特别地，next[1]=0

模式串：’aaaab’

序号j	1	2	3	4	5
模式串	a	a	a	a	b
next[j]	0	1	2	3	4

26

练习：求模式串的next数组

串的前缀：包含第一个字符，且不包含最后一个字符的子串
串的后缀：包含最后一个字符，且不包含第一个字符的子串
当第j个字符匹配失败，由前 1~j-1 个字符组成的串记为S，则：
next[j]= S的最长相等前后缀长度+1
特别地，next[1]=0

最长相等前后缀长度+1

$$next[j]=\begin{cases}0 & \text{当}j=1\text{时} \\ \text{Max}\{k \mid 1<k<j \text{且} 'p_1\cdots p_{k-1}' = 'p_{j-k+1}\cdots p_{j-1}'\} & \text{当此集合不空时} \\ 1 & \text{其他情况}\end{cases}$$

前缀　后缀　前后缀不匹配

27

KMP算法性能分析

```
//求模式串T的next数组
void get_next(SString T,int next[]){
    int i=1, j=0;
    next[1]=0;
    while(i<T.length){
        if(j==0||T.ch[i]==T.ch[j]){
            ++i; ++j;
            //若pi=pj，则 next[j+1]=next[j]+1
            next[i]=j;
        }
        else
            //否则令j=next[j]，循环继续
            j=next[j];
    }
}
```

KMP 算法平均时间复杂度：O(n+m)

```
//KMP算法
int Index_KMP(SString S,SString T){
    int i=1, j=1;
    int next[T.length+1];
    get_next(T, next);  //求模式串的next数组
    while(i<=S.length&&j<=T.length){
        if(j==0||S.ch[i]==T.ch[j]){
            ++i;
            ++j;                    //继续比较后继字符
        }
        else
            j=next[j];              //模式串向右移动
    }
    if(j>T.length)
        return i-T.length;          //匹配成功
    else
        return 0;
}
```

O(m)　O(n)

28

知识回顾与重要考点

朴素模式匹配算法的缺点：当某些子串与模式串能部分匹配时，主串的扫描指针 i 经常回溯，导致时间开销增加。最坏时间复杂度O(nm)

KMP算法：当子串和模式串不匹配时，主串指针 i 不回溯，模式串指针 j=next[j]
算法平均时间复杂度：O(n+m)

next数组手算方法：当第j个字符匹配失败，由前 1~j-1 个字符组成的串记为S，则：
next[j]=S的最长相等前后缀长度+1
特别地，next[1]=0

如果不会经常出现子串与模式串部分匹配问题，那么KMP算法也没屌多少

王道考研/CSKAOYAN.COM

29