

栈与递归实例

•递归调用的剖析

定义一个数x的非负整数n次幂的函数是递归函数的一个很好的例子。这个函数最直接的定义是：

$$x^n = \begin{cases} 1 & n=0 \\ x \cdot x^{n-1} & n>0 \end{cases}$$

可以直接根据幂的定义写出计算 x^n 的C++函数：

```
/*102*/ double power (double x, unsigned int n){  
/*103*/   if (n==0)  
/*104*/     return 1.0;  
           //else  
/*105*/   return x*power (x,n-1);};
```



改进的算法

- **KMP算法**
 - 无回溯算法
 - Knuth、Morris、Pratt等人发现



$$ts+j \dots$$


• • •

...

$$P_0 \sim \dots \sim P_{k-1}$$
$$p_0 \dots p_k \quad p_{k+1} \dots$$
$$p_0 \dots p_{k-1} \quad p_k$$

• • •

字符串

KMP算法

- 思想：当在某个位置匹配不成功的时候可以根据之前的匹配结果从模式字符串的另一个位置开始,而不必从头开始匹配字符串.

K值如何确定?

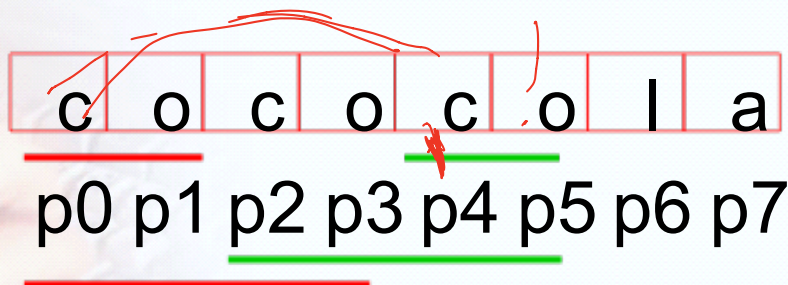
T
 $t_0 \dots t_s \quad t_{s+1} \quad t_{s+2} \quad t_{s+3} \dots t_{s+j-k-1} \quad t_{s+j-k} \dots t_{s+j-1}$
 $t_{s+j} \dots$

P
 $p_0 \quad p_1 \quad p_2 \quad p_3 \dots p_{j-k-1} \quad p_{j-k} \dots p_{j-1} \quad p_j$

$p_0 \dots p_{k-1} \quad p_k \dots$

预备知识

- 前缀子串：模式串P开头的前k个字符，
 p_0, p_1, \dots, p_{k-1} 。不包括尾巴
- i位置的后缀子串：在P第i位置的左边，取出k个字符，即 p_{i-k}, \dots, p_i 。包含i位置
- 第i位的最长前缀串：与i位置后缀子串相等的最长前缀子串。



模式串的特征数和特征向量

- 第 i 位的最长前缀串的长度 k 就是模式串 P 在位置 i 上的特征数 $n[i]$
- 特征数组成的向量称为该模式串的特征向量。
- 其意义在于：
 - 表示一旦匹配过程中 p_i 与 t_j 比较不等，可用 p 中以 $n[i-1]$ 为下标的字符与 t_j 进行比较。



特征数定义

$$n[j] = \left[\begin{array}{l} \text{MAX } \{ k, 0 < k \leq j \text{ 且 } P_0 \dots P_{k-1} = P_{j-k+1} \dots P_j - 1 \\ P_j \} \end{array} \right]$$

前缀串 = 后缀串
左串

0 其它情况 前缀串 = 后缀串

示例：确定特征数 $n[j]$ ， $P = \text{cococola}$

j	0	1	2	3	4	5	6	7
P	c	o	c	o	c	o	l	a
$n[j]$	0	0	1	2	3	4	0	0



特征数计算

- $n[0] = 0;$
- 设 $n[j] = k+1, 0 < k+1 \leq j$

$p_0 \ p_1 \ \dots \ p_{k-1} \ \quad p_k = \ p_{j-k} \ p_{j-k+1} \ \dots \ p_{j-1} \ \quad p_j$

等价于:

(1) $-1 < k \leq j-1$

(2) $p_0 \ p_1 \ \dots \ p_{k-1} \ = p_{j-k} \ p_{j-k+1} \ \dots \ p_{j-1}$

(3) $p_k = p_j$

字符串



特征数计算

对满足条件(1)(2)的k从大到小依次验证条件(3)是否成立:

$$\exists k1 = n[j-1] \quad p0 \dots p_{k1-1} = p_{j-k1} \dots p_{j-k2} \dots$$

$$p_{j-1} = n[k1-1]$$

$pk1 \neq pj; k2$

$= pk1 - k2 \dots$

$p0 \dots p_{k2-1} = p_{j-k2} \dots p_{j-1} \dots pk1-1$

$pk2 \neq pj+1;$

$p0 \dots p_{k1-1} = p_{j-k1} \dots p_{j-1}$

$pk = pj$

$pk1 \neq pj$

$p0 \dots pk2-1 = p_{j-k2} \dots p_{j-1}$

$p0 \dots p_{k1-1} = p_{j-k1} \dots p_{j-1}$

$k2 = n[k1-1]$

如果不存在满足条件的k, 则 $n[j]=0$;

$k2 = n[k1-1]$ 前面一串特征数

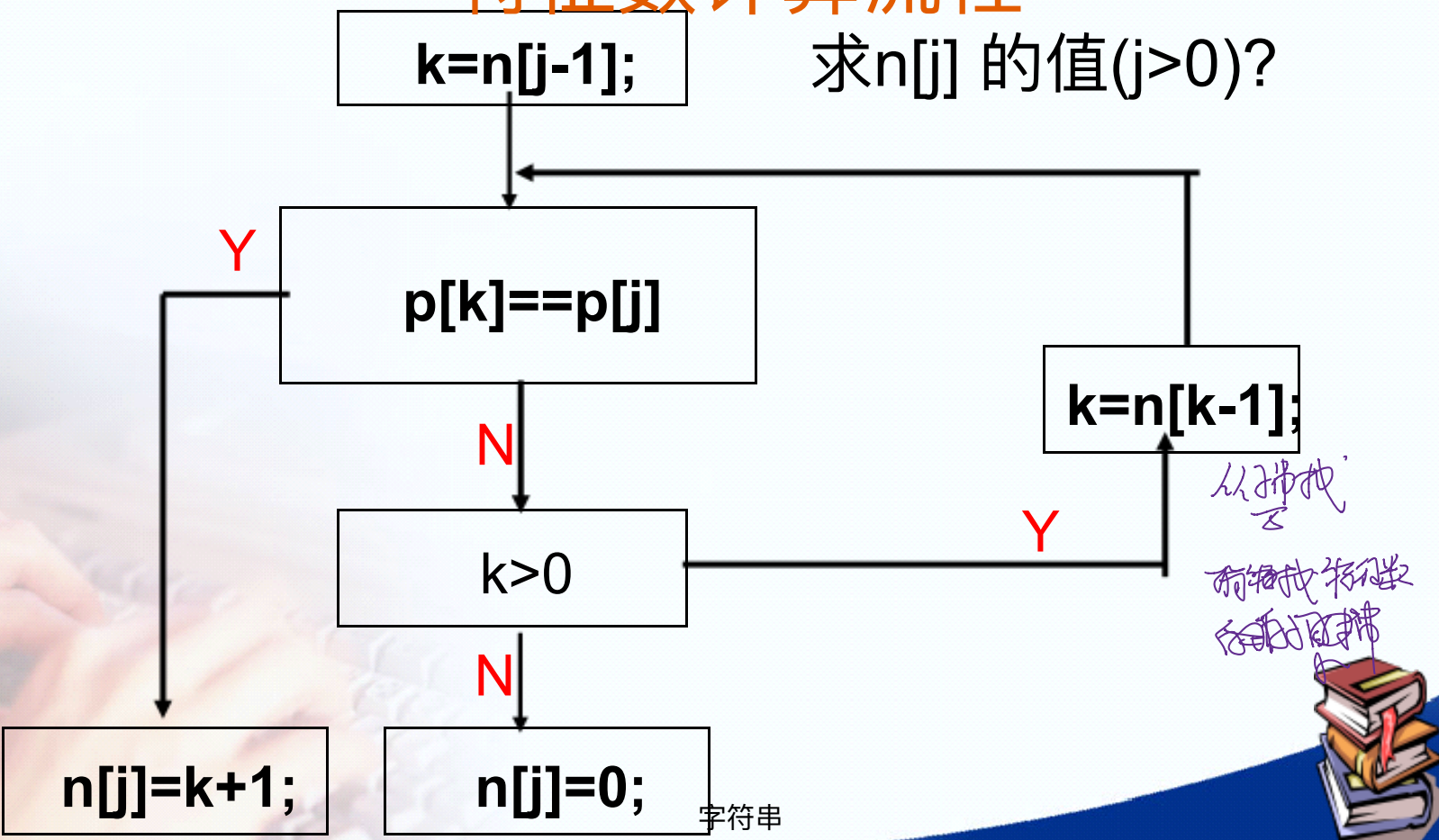


$P_{k+1} = P_j$

$k \leftarrow n[n[j]-1] \quad n[k+1]$
向左

特征数计算流程

求 $n[j]$ 的值($j>0$)?



KMP算法过程


(1) 预处理模式P，计算特征数


(2) 执行匹配过程



KMP算法匹配过程——case1


当前趟比较：

T $t_0 \dots ts$  $ts+1 \ ts+2 \ ts+3 \ \dots$


P p_0  $p_1 \ p_2 \ \dots$



下一趟比较：

T $t_0 \dots ts$  $ts+1 \ ts+2 \ ts+3$

\dots

P p_0  p_1 $p_2 \ \dots$

字符串



当前趟比较：

下一趟比较:

$$k = n(y-1)$$

其中： $k=n[j-1]$

字符串



KMP匹配过程

初始化 $i=0$; $j=0$;

$ti \neq '\backslash 0' \&\&$
 $pj \neq '\backslash 0'$ 未到最后

$j = n[j-1];$

$i = i + 1;$
同字符

$j == 0$

$ti == pj$

输出 $i-j$ 或 -1

$i++; j++;$

字符串

N

Y

Y

N

N

Y

↓
t0 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10

i	c	o	c	o	c	o	c	o	l	a
c	o	c	o	c	o	l	a			

p0 p1 p2 p3 p4 p5 p6 p7

× 目标指针加1

←———→

i	c	o	c	o	c	o	c	o	l	a
	c	o	c	o	c	o	l	a		

p0 p1 p2 p3 p4 p5 p6 p7

$k, n(j-1)$

× 模式指针 = $n[6-1]$

↓

i	c	o	c	o	c	o	c	o	l	a
		c	o	c	o	c	o	l	a	

p0 p1 p2 p3 p4 p5 p6 p7

字符串

14

第一趟:

第二趟:

第三趟:

KMP算法复杂度分析

- KMP匹配算法的时间复杂度 $O(\underline{m+n})$ 。 $n \gg m$
 $\approx O(n)$
- 计算特征数的时间复杂度为 $O(\underline{m})$
- 均摊分析



卡特兰数: Catalan

$$f(0)=1 \quad f(1)=1$$

$$f(n) = \sum_{i=0}^{n-1} f(i) f(n-i-1)$$

$$n \geq 2, \quad f(n) = \frac{C_{2n}}{n+1}$$

n 种出栈

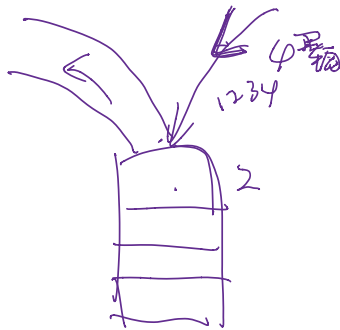
长

4

3

2

1



C_4

$$\frac{2 \times 7 \times 5}{4 \times 3 \times 2}$$

$$f(n) = \sum_{k=1}^n f(k-1) f(n-k)$$