KMP算法

汇报人：XXX

20XX.XX.XX

汇报提纲

- 一.KMP背景介绍
- 二.由朴素匹配到KMP
- 三.KMP核心——跳转表next[]
- 四. next[]的计算——引入f(j)

一.KMP背景介绍

在文本编辑中，我们经常要在一段文本中某个特定的位置找出某个特定的字符或模式。再比如，在HTTP协议里的字节流,有各种关键的字节流字段,对HTTP数据进行解释就需要用到模式匹配算法。由此，便产生了字符串的匹配问题。

KMP算法是由Knuth，Morris，Pratt三人共同提出的模式匹配算法，其对于任何模式和目标序列，都可以在线性时间内完成匹配查找，而不会发生退化，是一个非常优秀的模式匹配算法。

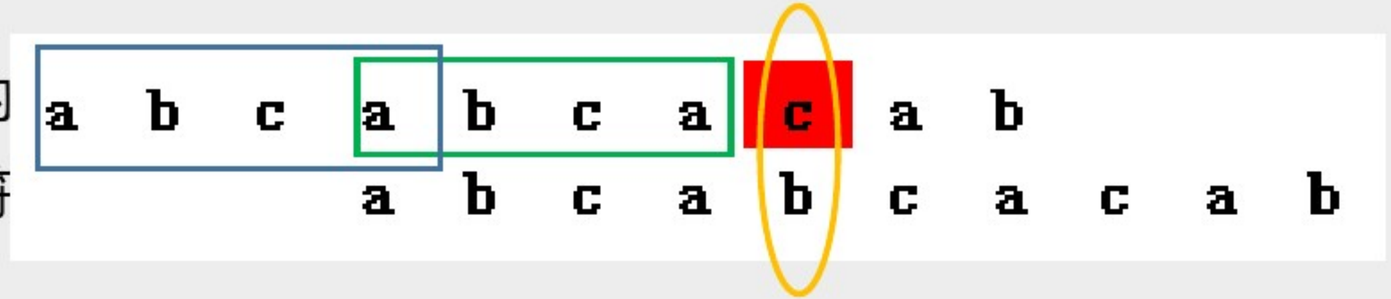
KMP算法

T角标j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
目标字符串（T）	b	a	b	c	b	a	b	c	a	b	c	a	a	b	c	a	b	c	a	b	c	a	c	a	b	c
模式字符串（P）	a	b	c	a	b	c	a	c	a	b																
		a	b	c	a	b	c	a	c	a	b															
			a	b	c	a	b	c	a	c	a	b														
				a	b	c	a	b	c	a	c	a	b													
					a	b	c	a	b	c	a	c	a	b												
						a	b	c	a	b	c	a	c	a	b											
							a	b	c	a	b	c	a	c	a	b										
								a	b	c	a	b	c	a	c	a	b									
									a	b	c	a	b	c	a	c	a	b								
										a	b	c	a	b	c	a	c	a	b							
											a	b	c	a	b	c	a	c	a	b						
												a	b	c	a	b	c	a	c	a	b					
													a	b	c	a	b	c	a	c	a	b				
														a	b	c	a	b	c	a	c	a	b			
															a	b	c	a	b	c	a	c	a	b		
																a	b	c	a	b	c	a	c	a	b	

朴素匹配的时间复杂度为**O(m*n)**；
KMP的时间复杂度为**O(n)**。

由朴素匹配，我们要做**16次**，而KMP算法仅匹配了**6次**就找到了模式字符串

三.KMP算法核心——跳转表next[]



其实，模式串往往含有一定的

对于模式串而言，其前缀字符

我称之为前缀包含问题。

以模式串a b c a b c a c a b为例，其前缀的4个字符a b c a，正好也是模式串的一个子串a b c (a b c a) **c** a b，所以当目标串与模式串执行匹配的过程中，如果直到第 8 个字符才匹配失败，同时也意味着目标串当前字符之前的 4 个字符，与模式串的前 4 个字符是相同的，所以当模式串向后移动的时候，可以直接将模式串的第 5 个字符与当前字符对齐，执行比较，这样就实现了模式串一次性向前跳跃多个字符。

那么每次要跳跃多少呢？这与跳转表next[]存储的数值相关。

三.KMP算法核心——跳转表next[]

模式字符串P= a b c a b c a c a b，其跳转表为：

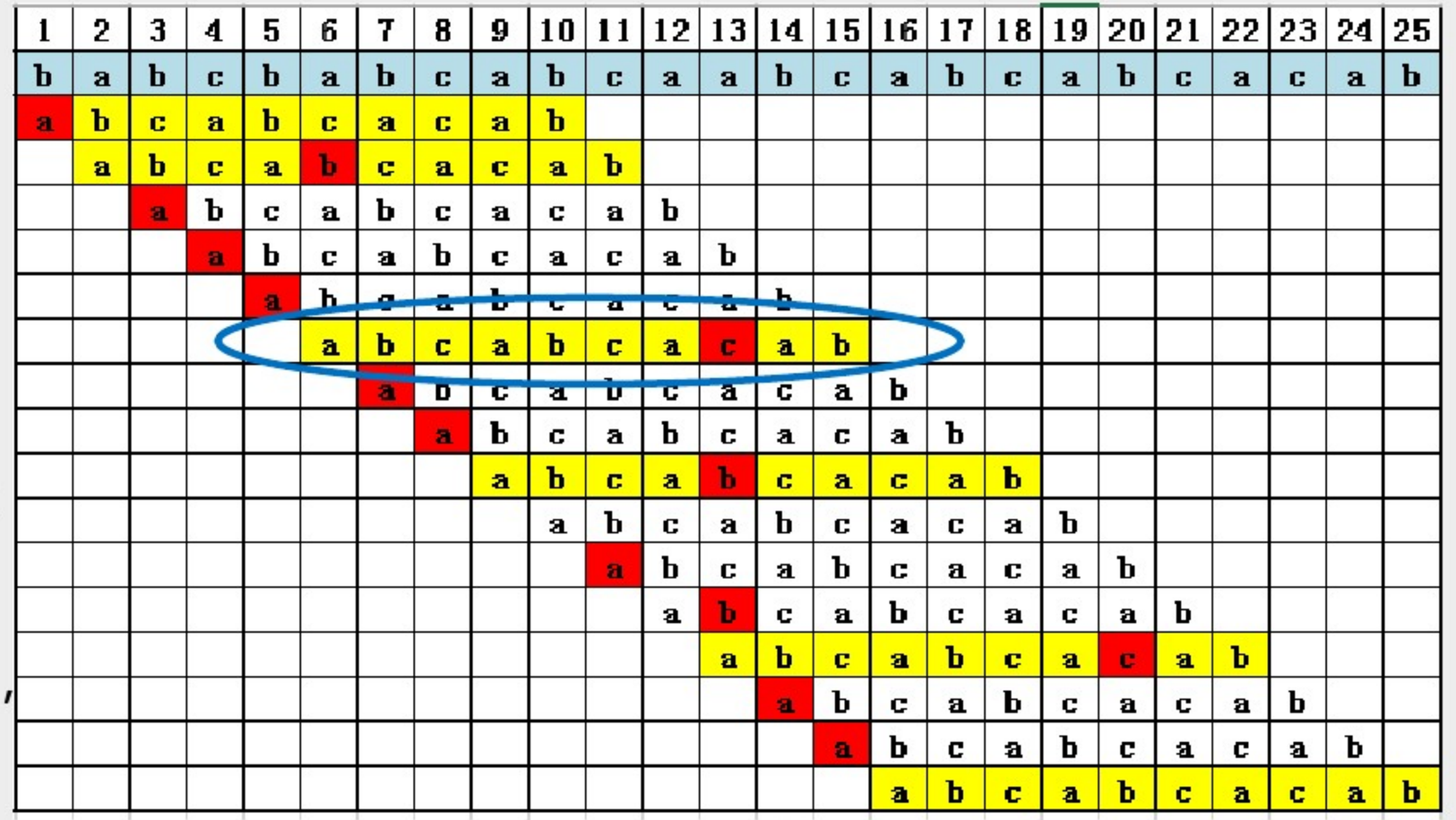
j	1	2	3	4	5	6	7	8	9	10
pattern[j]	a	b	c	a	b	c	a	c	a	b
next[j]	0	1	1	0	1	1	0	5	0	1

j	1	2	3	4	5	6	7	8	9	10
pattern[j]	a	b	c	a	b	c	a	c	a	b
next[j]	0	1	1	0	1	1	0	5	0	1

三.KMP算法核心——跳转表next[]

我们以KMP匹配的第 3 步为例:

此时 pattern 串的第 1 个字母与 target[6]对齐，从 6 向后依次匹配目标串，到 target[13]时发现 target[13]='a'，而 pattern[8]='c'，匹配失败，此时 next[8]=5，所以将模式串向后移动 8-next[8] = 3 个字符，将pattern[5]与 target[13]对齐，然后由 target[13]依次向后执行匹配操作。在整个匹配过程中，无论模式串如何向后滑动，目标串的输入字符都不会回溯，直到找到模式串，或者遍历整个目标串都没有发现匹配模式为止。



四. next[]的计算——引入f(j)

跳转表next[]是如何计算的呢？以及怎样以较小的代价计算?

这里我们引入一个概念 f(j)，其含义是，对于模式串的第 j 个字符 pattern[j]，f(j)是所有满足使 pattern[1…k-1] =pattern[j-(k-1)…j-1] (k < j) 成立的 k 的最大值。**f(j)=k**

我们可以看出k最小为2，

因此，规定f(1)=0；

不存在前缀包含时，f(j)=1

模式字符串P= a b c a b c a c a b的f(j)计算结果如下：

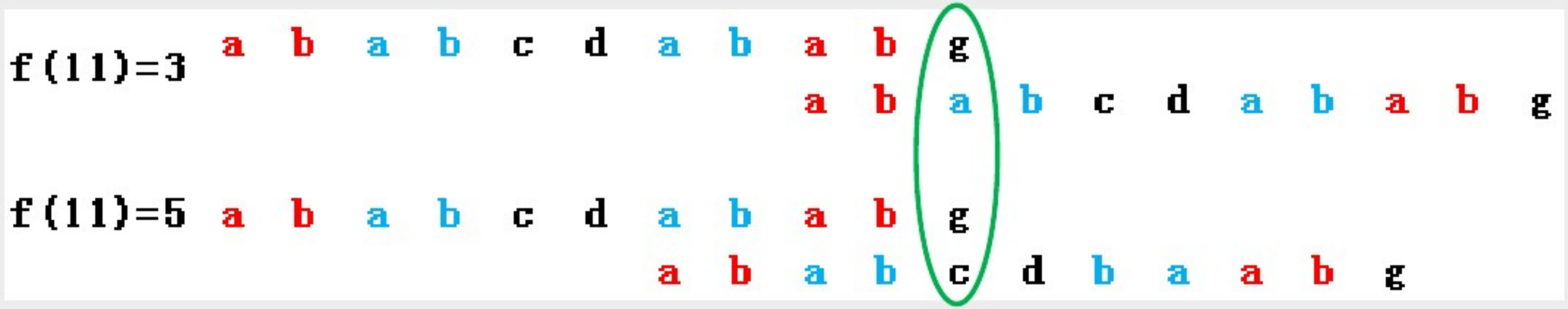
j	1	2	3	4	5	6	7	8	9	10
pattern[j]	a	b	c	a	b	c	a	c	a	b
next[j]	0	1	1	0	1	1	0	5	0	1
f(j)	0	1	1	1	2	3	4	5	1	2

四. next[]的计算——引入f(j)

f(j) 含义：对于模式串的第 j 个字符 pattern[j]，f(j)是所有满足使 pattern[1…k-1] =pattern[j-(k-1)…j-1] (k < j) 成立的 k 的最大值。**f(j)=k**

如何理解取K最大值呢？

比如，假设一个11位模式字符串为a b a b c d a b a b **g**，则f(11)=5 != 3。



通过上图，我们不难看出，k越小，跳跃的步伐越大，很可能会把满足条件的匹配结果跳过去，因此我们在保证算法快速的同时，还要保证准确！

四. next[]的计算——引入f(j)

j	1	2	3	4	5	6	7	8	9	10
pattern[j]	a	b	c	a	b	c	a	c	a	b
next[j]	0	1	1	0	1	1	0	5	0	1
f(j)	0	1	1	1	2	3	4	5	1	2

为了说明f(j)和next[j]之间的关系，我们以pattern[8]为例，假如匹配到pattern[8]才匹配失败。f(8)=5,pattern[1…4]=pattern[4…7]，此时我们需要关注pattern[8]：

1. 如果pattern[8] != pattern[5]

因为在匹配到 pattern[8]时才失败，此时就可以将输入字符 target[n]与 pattern[f(8)]=pattern[5]对齐，再向后依次执行匹配，所以此时的 next[8] = f(8)。

四. next[]的计算——引入f(j)

j	1	2	3	4	5	6	7	8	9	10
pattern[j]	a	b	c	a	b	c	a	c	a	b
next[j]	0	1	1	0	1	1	0	5	0	1
f(j)	0	1	1	1	2	3	4	5	1	2

2. 如果pattern[8] = pattern[5]

那么pattern[1…5]=pattern[4…8]，因为target[n]与 pattern[8]匹配失败，那么也意味着target[n-4…n]!=pattern[4…8]，那么将 target[n]与 pattern[5]对齐， target[n-4…n]也必然不等于pattern[1…5]。

此时我们需要关注f(5)。

四. next[]的计算——引入f(j)

j	1	2	3	4	5	6	7	8	9	10
pattern[j]	a	b	c	a	b	c	a	c	a	b
next[j]	0	1	1	0	1	1	0	5	0	1
f(j)	0	1	1	1	2	3	4	5	1	2

2. 如果pattern[8] = pattern[5]

f(5) = 2，这意味着 pattern[1] = pattern[4]，因为pattern[1…4]=pattern[4…7]，所以pattern[4]=pattern[7]=pattern[1],此时我们再来比较pattern[8]和pattern[2]。

2.1如果pattern[8] != pattern[2]

就可以将target[2]与pattern[2]对齐，然后比较二者是否相等，此时

next[8]= next[f(8)]= next[5]=f(5)。

四. next[]的计算——引入f(j)

j	1	2	3	4	5	6	7	8	9	10
pattern[j]	a	b	c	a	b	c	a	c	a	b
next[j]	0	1	1	0	1	1	0	5	0	1
f(j)	0	1	1	1	2	3	4	5	1	2

2.2如果pattern[8] = pattern[2]

那么还需要考察pattern[f(2)] ，

……

直到回溯到模式串头部为止。

四. next[]的计算——引入f(j)

j	1	2	3	4	5	6	7	8	9	10
pattern[j]	a	b	c	a	b	c	a	c	a	b
next[j]	0	1	1	0	1	1	0	5	0	1
f(j)	0	1	1	1	2	3	4	5	1	2

由以上分析，我们可以归纳出根据 f(j) 求next[j]的递推公式：

如果 pattern[j] != pattern[f(j)]， next[j] = f(j);

如果 pattern[j] = pattern[f(j)]， next[j] = next[f(j)];

j	1	2	3	4	5	6	7	8	9	10
pattern[j]	a	b	c	a	b	c	a	c	a	b
next[j]	0	1	1	0	1	1	0	5	0	1
f(j)	0	1	1	1	2	3	4	5	1	2

T角标j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
目标字符串（T）	b	a	b	c	b	a	b	c	a	b	c	a	a	b	c	a	b	c	a	b	c	a	c	a	b	c
模式字符串（P）	a	b	c	a	b	c	a	c	a	b																
		a	b	c	a	b	c	a	c	a	b															
			a	b	c	a	b	c	a	c	a	b														
				a	b	c	a	b	c	a	c	a	b													
					a	b	c	a	b	c	a	c	a	b												
						a	b	c	a	b	c	a	c	a	b											
							a	b	c	a	b	c	a	c	a	b										
								a	b	c	a	b	c	a	c	a	b									
									a	b	c	a	b	c	a	c	a	b								
										a	b	c	a	b	c	a	c	a	b							
											a	b	c	a	b	c	a	c	a	b						
												a	b	c	a	b	c	a	c	a	b					
													a	b	c	a	b	c	a	c	a	b				
														a	b	c	a	b	c	a	c	a	b			
															a	b	c	a	b	c	a	c	a	b		
																a	b	c	a	b	c	a	c	a	b	
																	a	b	c	a	b	c	a	c	a	b

谢 谢

汇报人：XXX

学　号：XXXXXXXX

班　级：XXXXX