**看毛片算法next数组求解**

在字符串处理中常常会有字符串匹配的操作，暴力搜索的时间复杂度是o(mn)，而KMP算法的时间复杂度是o(m+n),很明显KMP算法在时间上会有优势。我们都知道数据结构与算法中，总是在时间和空间上取得平衡，有时候会牺牲时间来取得空间效率上的提高，也有时候会选择牺牲空间来换取时间效率上面的提高，而KMP算法则是选择了后者，在内存中开辟一段空间，来存放字符串每个字符对应的部分匹配值（next数组）。个人觉得吧KMP算法的核心就是求这个next数组，但是不是很好理解，下面就来解释下next数组。

第一种求next数组的方法：

字符串：abcaab

a ： 0

ab ：前缀［a］

后缀［b］

无共有元素

abc ：前缀［a］［ab］

后缀［c］［bc］

无共有元素

abca ：前缀［a］［ab］［abc］

后缀［a］［ca］［bca］

共有元素：［a］长度为1

abcaa ：前缀［a］［ab］［abc］［abca］

后缀［a］［aa］［aac］［aacb］

共有元素：［a］长度为1

abcaab：前缀［a］［ab］［abc］［abca］［abcaa］

后缀［b］［ab］［aab］［caab］［bcaab］

共有元素：［ab］长度为2

上述方法个人觉得过于麻烦，下面介绍一种比较直观也比较好理解的方法：

直接观察字符串：



1. abcaab这个字符串开头字母为（0位置）a 向后找找到一个（3位置）a 则这个a的next数组值则为1。



1. 开始对比第二个字符，显然（1位置）b与（4位置）a不相等。



1. 退回开头字母（0位置）a，显然（0位置）a与（4位置）a相等则（4位置）的a next数组值为1。



1. 接着对比第二个字符（1位置）b与（5位置）b相等，则（5位置）b对应的next数组值为前一个字符（4位置）next数组值＋1，即2。

数组使用公式：移动位数 = 已匹配的字符数 - 对应的部分匹配值

下面来看一下这个数组的具体使用方法：



字符串abcabcaabc与字符串abcaab进行匹配



对应位置a的部分匹配值为1，则移动位数为4-1=3得：



这样next数组的功能一目了然了吧。

代码的实现思路与上述方法一致，下面贴出代码：

