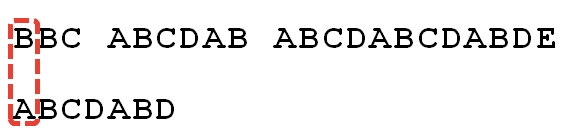
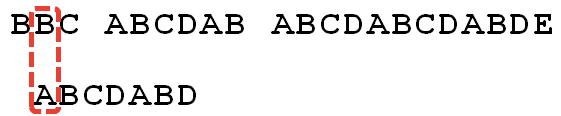
举例来说，有一个字符串 Str1 = “BBC ABCDAB ABCDABCDABDE”，判断，里面是否包含另一个字符串 Str2 = “ABCDABD”？

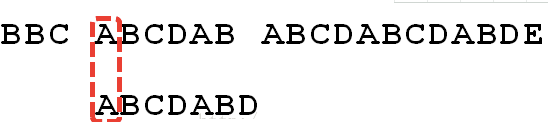
1.首先，用Str1的第一个字符和Str2的第一个字符去比较，不符合，关键词向后移动一位



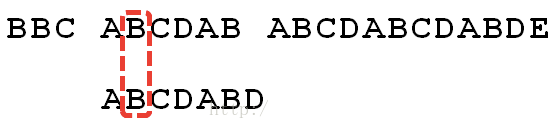
1. 重复第一步，还是不符合，再后移



1. 一直重复，直到Str1有一个字符与Str2的第一个字符符合为止



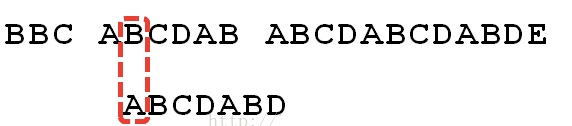
1. 接着比较字符串和搜索词的下一个字符，还是符合。



5.遇到Str1有一个字符与Str2对应的字符不符合。



6.这时候，想到的是继续遍历Str1的下一个字符，重复第1步。(其实是很不明智的，因为此时BCD已经比较过了，没有必要再做重复的工作，一个基本事实是，当空格与D不匹配时，你其实知道前面六个字符是”ABCDAB”。KMP 算法的想法是，设法利用这个已知信息，不要把”搜索位置”移回已经比较过的位置，继续把它向后移，这样就提高了效率。)



7.怎么做到把刚刚重复的步骤省略掉？可以对Str2计算出一张《部分匹配表》，这张表的产生在后面介绍

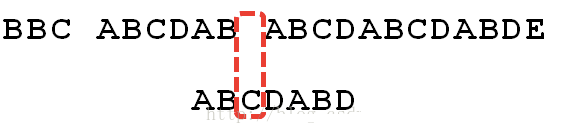


8.已知空格与D不匹配时，前面六个字符”ABCDAB”是匹配的。查表可知，最后一个匹配字符B对应的”部分匹配值”为2，因此按照下面的公式算出向后移动的位数：

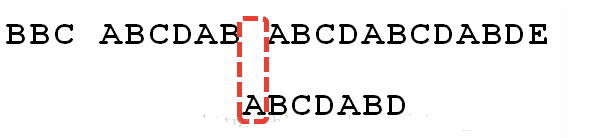
移动位数 = 已匹配的字符数 - 对应的部分匹配值

因为 6 - 2 等于4，所以将搜索词向后移动 4 位。

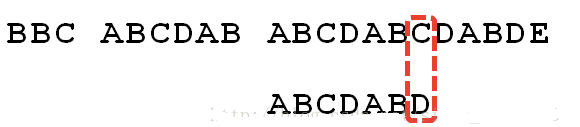
9.因为空格与Ｃ不匹配，搜索词还要继续往后移。这时，已匹配的字符数为2（”AB”），对应的”部分匹配值”为0。所以，移动位数 = 2 - 0，结果为 2，于是将搜索词向后移 2 位。



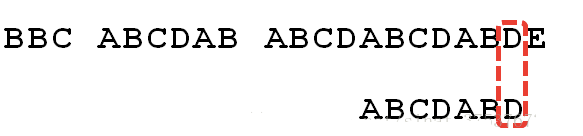
10.因为空格与A不匹配，继续后移一位。



11.逐位比较，直到发现C与D不匹配。于是，移动位数 = 6 - 2，继续将搜索词向后移动 4 位。

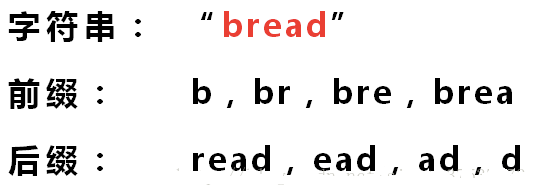


12.逐位比较，直到搜索词的最后一位，发现完全匹配，于是搜索完成。如果还要继续搜索（即找出全部匹配），移动位数 = 7 - 0，再将搜索词向后移动 7 位，这里就不再重复了。



13.介绍《部分匹配表》怎么产生的

先介绍前缀，后缀是什么



“部分匹配值”就是”前缀”和”后缀”的最长的共有元素的长度。以”ABCDABD”为例，

－”A”的前缀和后缀都为空集，共有元素的长度为0；

－”AB”的前缀为[A]，后缀为[B]，共有元素的长度为0；

－”ABC”的前缀为[A, AB]，后缀为[BC, C]，共有元素的长度0；

－”ABCD”的前缀为[A, AB, ABC]，后缀为[BCD, CD, D]，共有元素的长度为0；

－”ABCDA”的前缀为[A, AB, ABC, ABCD]，后缀为[BCDA, CDA, DA, A]，共有元素为”A”，长度为1；

－”ABCDAB”的前缀为[A, AB, ABC, ABCD, ABCDA]，后缀为[BCDAB, CDAB, DAB, AB, B]，共有元素为”AB”，长度为2；

－”ABCDABD”的前缀为[A, AB, ABC, ABCD, ABCDA, ABCDAB]，后缀为[BCDABD, CDABD, DABD, ABD, BD, D]，共有元素的长度为0。

14.”部分匹配”的实质是，有时候，字符串头部和尾部会有重复。比如，”ABCDAB”之中有两个”AB”，那么它的”部分匹配值”就是2（”AB”的长度）。搜索词移动的时候，第一个”AB”向后移动 4 位（字符串长度-部分匹配值），就可以来到第二个”AB”的位置。



到此KMP算法思想分析完毕!