软件设计师
--数据结构基础

高级项目经理　任铄
QQ：1530841586

第一章 数据结构基础

- 1.1 线性表
- 1.2 树和二叉树
- 1.3 图
- 1.4 排序
- 1.5 查找

向上人生路！

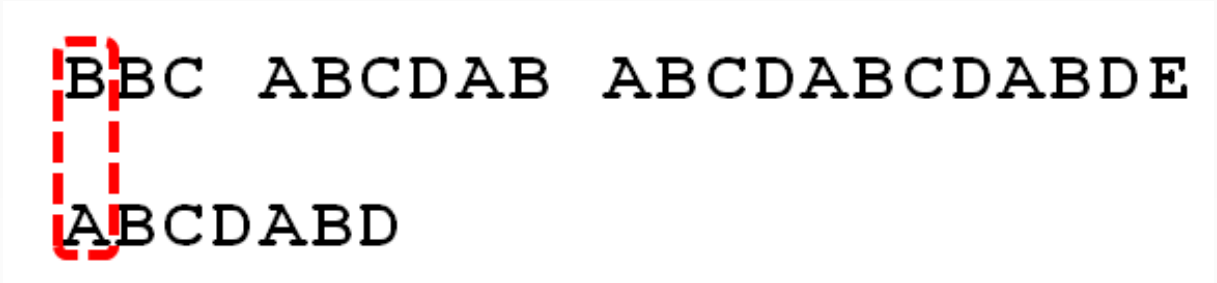
高级项目经理　任铄
QQ：1530841586

五、 KMP算法

- KMP算法是一种改进的字符串匹配算法。
- KMP算法的关键是利用匹配失败后的信息，尽量减少模式串与主串的匹配次数以达到快速匹配的目的。具体实现就是实现一个next()函数，函数本身包含了模式串的局部匹配信息。

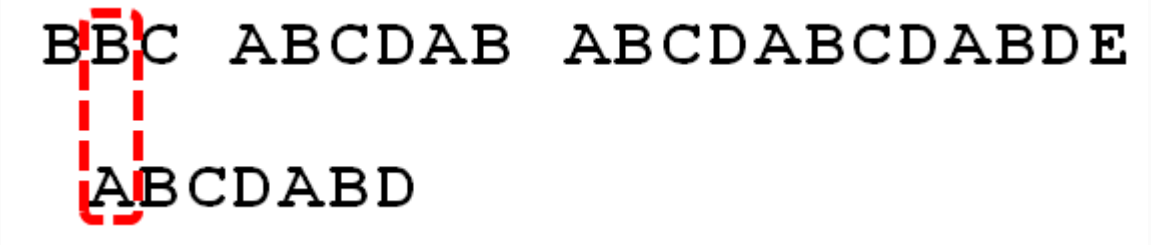
例：字符串"BBC ABCDAB ABCDABCDABDE"与搜索词"ABCDABD"的匹配。

1、



因为B与A不匹配，所以搜索词后移一位。

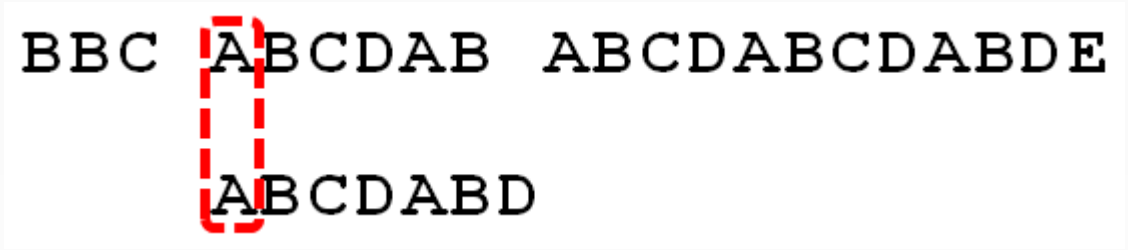
2、



因为B与A不匹配，搜索词再往后移。

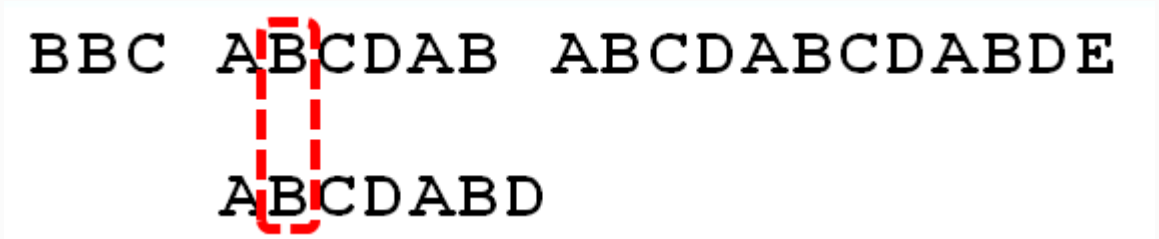
高级项目经理　任铄
QQ：1530841586

3、



直到字符串有一个字符，与搜索词的第一个字符相同为止。

4、

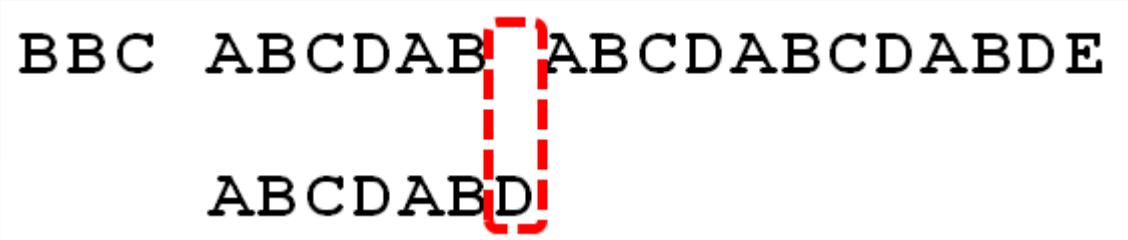


接着比较字符串和搜索词的下一个字符，还是相同。

高级项目经理　任钭
QQ：1530841586

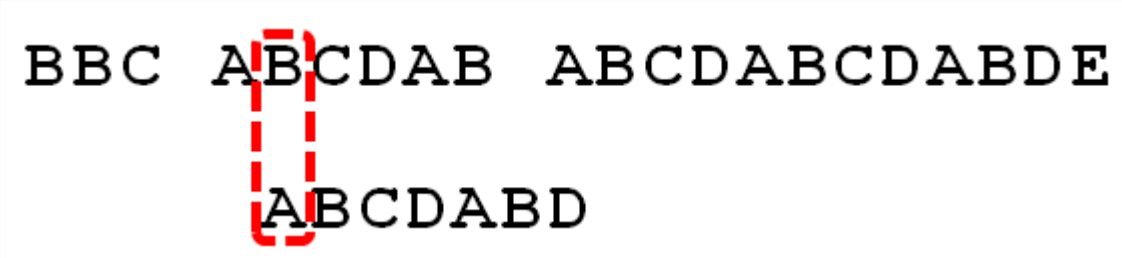
向上人生路！

5、



直到字符串有一个字符，与搜索词对应的字符不相同为止。

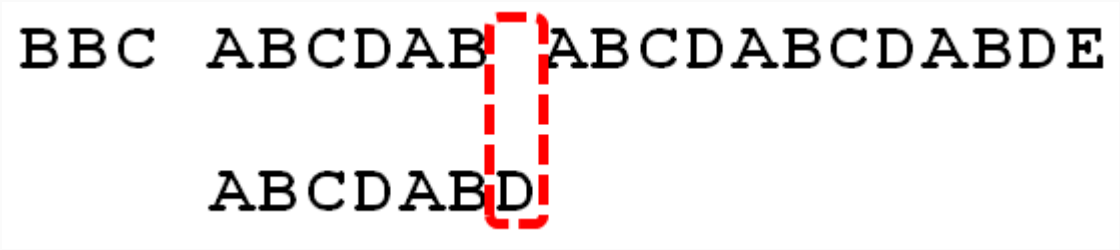
6、



这样做虽然可行，但是效率很差。

高级项目经理　任铄
QQ：1530841586

7、



KMP算法的想法是，利用已经知道的前面六个字符"ABCDAB"，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。

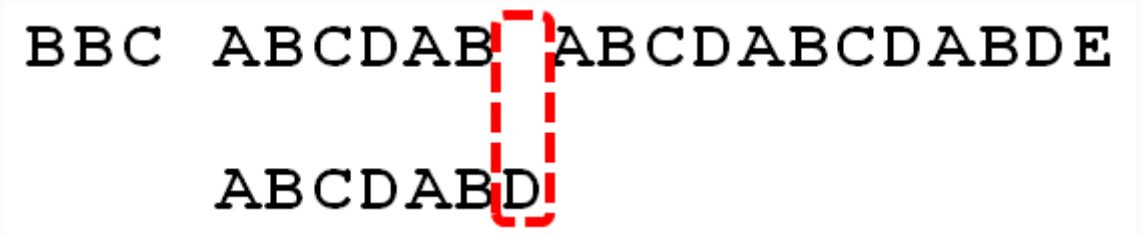
高级项目经理　任钭
QQ：1530841586

8、

搜索词	A	B	C	D	A	B	D
部分匹配值	0	0	0	0	1	2	0

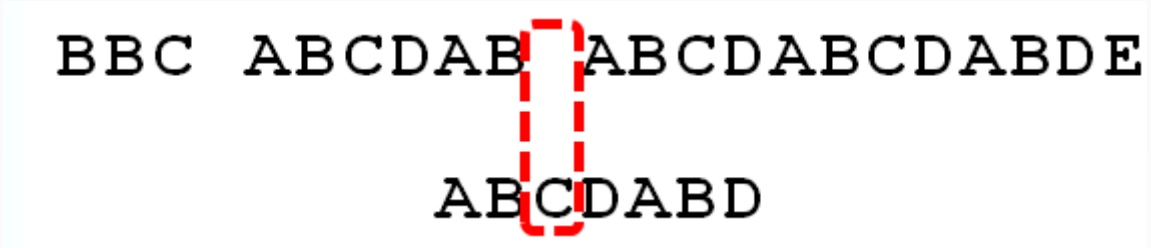
可以针对搜索词，算出一张《部分匹配表》

9、



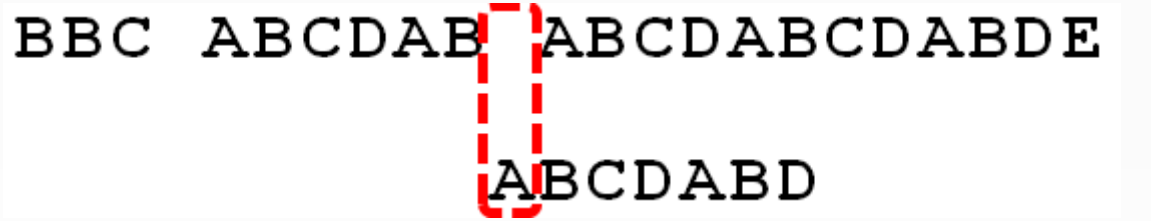
查表可知，最后一个匹配字符B对应的"部分匹配值"为2，因此按照公式（移动位数 = 已匹配的字符数 - 对应的部分匹配值） 算出向后移动的位数 6-2=4，将搜索词向后移动4位。

10、



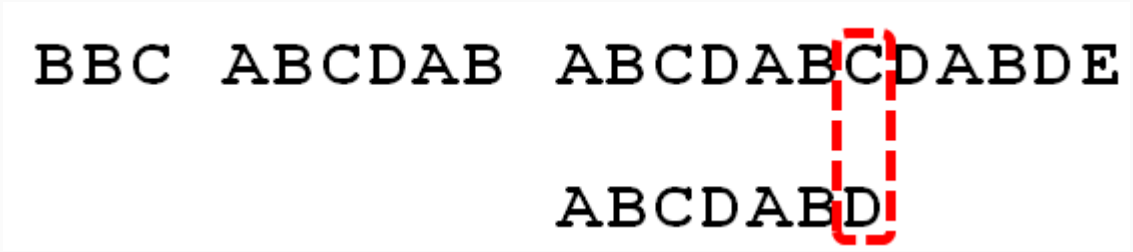
因为空格与C不匹配，搜索词还要继续往后移。这时，已匹配的字符数为2（"AB"），对应的"部分匹配值"为0。所以，移动位数 = 2 - 0，结果为 2，于是将搜索词向后移2位

11、



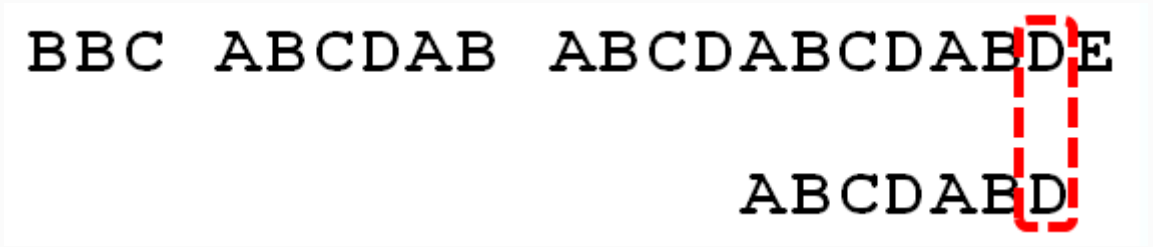
空格与A不匹配，继续后移一位。

12、



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。

13、



发现完全匹配，于是搜索完成

高级项目经理　任铄
QQ：1530841586

《部分匹配表》是如何产生的

首先了解两个概念："前缀"和"后缀"。

- "前缀"指除了最后一个字符以外，一个字符串的全部头部组合；
- "后缀"指除了第一个字符以外，一个字符串的全部尾部组合。

"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，

高级项目经理　任钭
QQ：1530841586

- "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。

高级项目经理　任铄
QQ：1530841586

搜索词	A	B	C	D	A	B	D
部分匹配值	0	0	0	0	1	2	0

例：在字符串的KMP模式匹配算法中，需先求解模式串p的next函数值，其定义如下。若模式串p为“abaabaca”，则其next函数值为（ B ）

$$next[j]=\begin{cases}0 & j=1\\ \max\{k \mid 1<k<j,\ 'p_1p_2\cdots p_{k-1}'='p_{j-k+1}p_{j-k+2}\cdots p_{j-1}'\} & \\ 1 & \text{其他情况}\end{cases}$$

A．01111111　　B．01122341

C．01234567　　D．01122334

高级项目经理　任钭
QQ：1530841586

解：

给定的字符串叫做模式串T。j表示next函数的参数，其值是从1到n。而k则表示一种情况下的next函数值。p表示其中的某个字符，下标从1开始。看等式左右对应的字符是否相等。

j	1	2	3	4	5	6	7	8
模式串T	a	b	a	a	b	a	c	a
next[j]								

高级项目经理　任钭
QQ：1530841586

1、j=1时，next[1]=0；

$$next[j]=\begin{cases}0 & j=1\\ \max\{k \mid 1<k<j,'p_1p_2\cdots p_{k-1}'='p_{j-k+1}p_{j-k+2}\cdots p_{j-1}'\} & \\ 1 & \text{其他情况}\end{cases}$$

2、j=2时，k的取值为(1,j)的开区间，所以整数k是不存在的，那就是第三种情况，next[2]=1；

3、j=3时，k的取值为（1，3）的开区间，k从最大的开始取值，然后带入含p的式子中验证等式是否成立，不成立k取第二大的值。现在是k=2，将k导入p的式子中得，p1=p2，即“a”=“b”，显然不成立，舍去。k再取值就超出范围了，所以next[3]不属于第二种情况，那就是第三种了，即next[3]=1；

高级项目经理　任铄
QQ：1530841586

$$next[j]=\begin{cases}0 & j=1\\ \max\{k \mid 1<k<j,'p_1p_2\cdots p_{k-1}'='p_{j-k+1}p_{j-k+2}\cdots p_{j-1}'\} & \\ 1 & \text{其他情况}\end{cases}$$

4、j=4时，k的取值为（1，4）的开区间，先取k=3，将k导入p的式子中得，p1p2=p2p3，不成立。 再取k=2，得p1=p3，成立。所以next[4]=2；

5、j=5时，k的取值为（1，5）的开区间，先取k=4，将k导入p的式子中得，p1p2p3=p2p3p4，不成立。 再取k=3，得p1p2=p3p4，不成立。 再取k=2，得p1=p4，成立。所以next[5]=2；

高级项目经理　任铄
QQ：1530841586

$$next[j]=\begin{cases}0 & j=1\\ \max\{k \mid 1<k<j,'p_1p_2\cdots p_{k-1}'='p_{j-k+1}p_{j-k+2}\cdots p_{j-1}'\} & \\ 1 & \text{其他情况}\end{cases}$$

6、j=6时，k的取值为（1，6）的开区间，先取k=5，将k导入p的式子中得，p1p2p3p4=p2p3p4p5，不成立。 取k=4，得p1p2p3=p3p4p5，不成立。再取k=3，将k导入p的式子中得，p1p2=p4p5，成立。所以next[6]=3；

7、j=7时，k的取值为（1，7）的开区间，先取k=6，将k导入p的式子中得，p1p2p3p4p5=p2p3p4p5p6，不成立。 再取k=5，得 p1p2p3p4=p3p4p5p6 ，不成立。 再取k=4，得p1p2p3=p4p5p6 ，成立。所以next[7]=4；

$$next[j]=\begin{cases}0 & j=1\\ \max\{k \mid 1<k<j,\text{'}p_1p_2\cdots p_{k-1}\text{'}=\text{'}p_{j-k+1}p_{j-k+2}\cdots p_{j-1}\text{'}\} & \\ 1 & \text{其他情况}\end{cases}$$

8、j=8时，k的取值为（1，8）的开区间， 先取k=7，将k导入p的式子中得，p1p2p3p4p5p6=p2p3p4p5p6p7，不成立。 再取k=6，得p1p2p3p4p5=p3p4p5p6p7，不成立。…… 再取k=2，得p1=p7，不成立。k再取值就超出范围了，所以next[8]不属于第二种情况，那就是第三种了，即next[8]=1；

j	1	2	3	4	5	6	7	8
模式串T	a	b	a	a	b	a	c	a
next[j]	0	1	1	2	2	3	4	1

可以通过下列渠道沟通联系：

1、 QQ:1530841586

2、 QQ群：164955673

向上人生路！