BM 算法详细图解

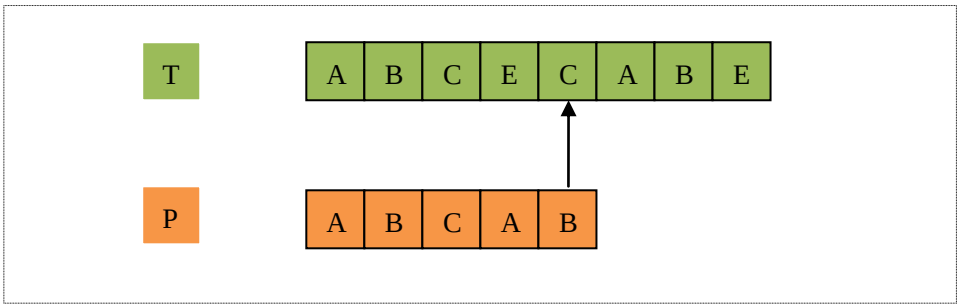
BM 算法（全称 Boyer-Moore Algorithm）是一种精确字符串匹配算法（只是一个启发式的字符串搜索算法）。

BM 算法不同于 KMP 算法，采用从右向左比较的方法，同时引入了两种启发式 Jump 规则，即 **Bad-Character** 和 **Good-Suffix**，来决定模板向右移动的步长。

BM 算法的基本流程:

文本串 T（待匹配的字符串，长度 n），模式串为 P（用于去匹配的字符串模板，长度设为 m）。

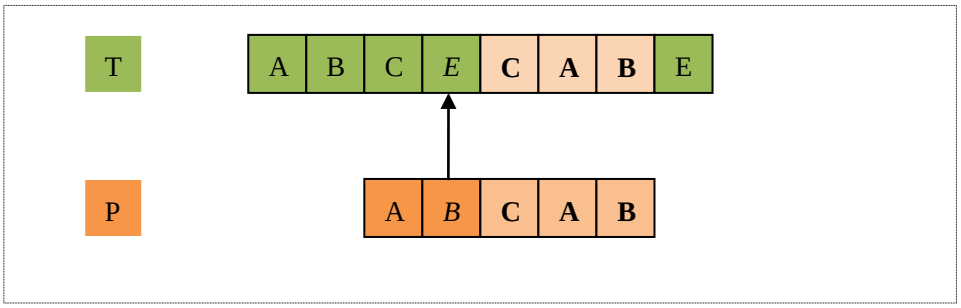
首先将 T 与 P 进行左对齐，然后进行从右向左比较（此时不移动模板和文本的相对位置，仅仅比较对齐位置上面是否相同，方向是右向左），如下图所示:



若是某趟比较不匹配时，BM 算法就采用两条启发式规则，即前面提到的 Bad-Character 和 Good-Suffix，来计算模式串 P 向右移动的步长（取两种方式求得的移动步长的较大的），直到整个匹配过程的结束。

下面，来详细介绍一下 Bad-Character 和 Good-Suffix：

请看下图：



图中，第一个不匹配的字符箭头连接的 E 和 B(斜体)为坏字符 Bad-Character，已匹配部分 CAB（加粗）为 Good-Suffix。

为了直观的理解 BM 算法，下面的详细步骤讲解利用上图具体说明：

1）Bad-Character

BM 算法在上图中从右向左匹配中出现不一致的情况，此时按照 Bad-Character 需要采用两种情况来处理：

a) 如果 T 中不匹配字符 E 在模式 P 中没有出现，那么我们很容易就能理解为 E 开始的 m 长度的字符串不可能匹配到 P（直观，无需解释），我们可以直接把 P 跳过 E，匹配后面的内容。
b) 如果 E 在模式 P 中未进行匹配的字段中出现了，则以该字符 E 进行对齐。

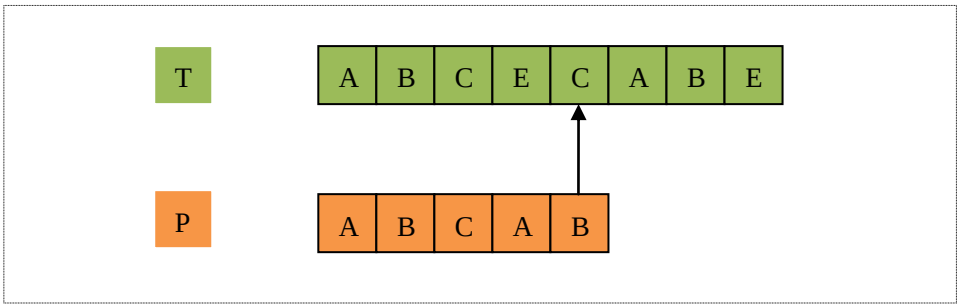
经过两次调整后，下面再次进行从右往左的匹配过程了。

用数学公式表示，设 Jump(x)为 P 右移的距离，m 为模式串 P 的长度，max(x)为字符 x 在 P 中出现的最右位置（字符串的索引编号从 1 开始）：

$$Jump(x) = \begin{cases} m - Len(x) & (\text{x不在P中未匹配的字段中出现}) \\ m - Last(\text{x}) & (\text{Last(x)代表在P中未匹配字段中出现的最后位置}) \end{cases}$$

其中 Ien(x)代表已经匹配字段的长度。

下面进行详细说明：



根据上述原则计算 Jump(C)步长：

此时 P 字符串的长度：

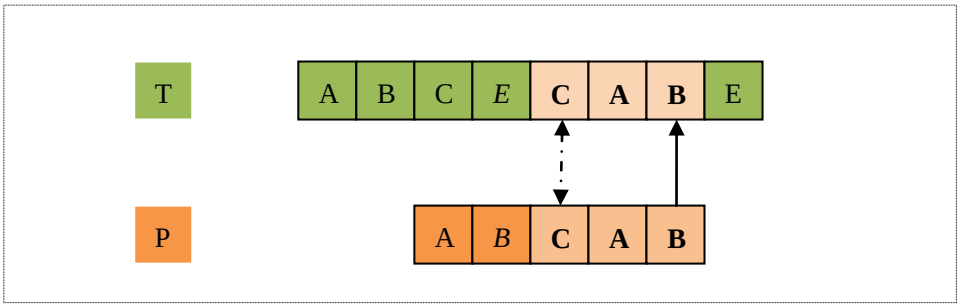
strlen(P)=5　模式串长度 5

Len(C)=0　已经匹配的长度为 0

Last(C)=3　C 出现的最后一个位置 3

Jump(C)=m-Len(C)=5-3=2

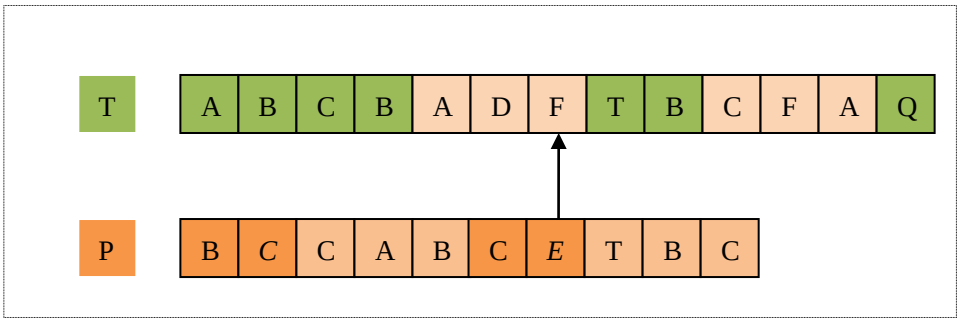
所以根据 Jump 的计算结果为移动 2，移动后的结果为下图：



这样移动的目的就是让 T 和 P 中正在匹配的字段中相同的字符对齐。

2）Good Suffix

若发现某个字符不匹配的同时，已有部分字符匹配成功，则按如下两种情况进行讨论：



上图中是之前分析中到的位置，匹配过程中在 C 处出现了不匹配现象，下面到底要怎么处理呢，Shift(x)应该是移动多少呢？

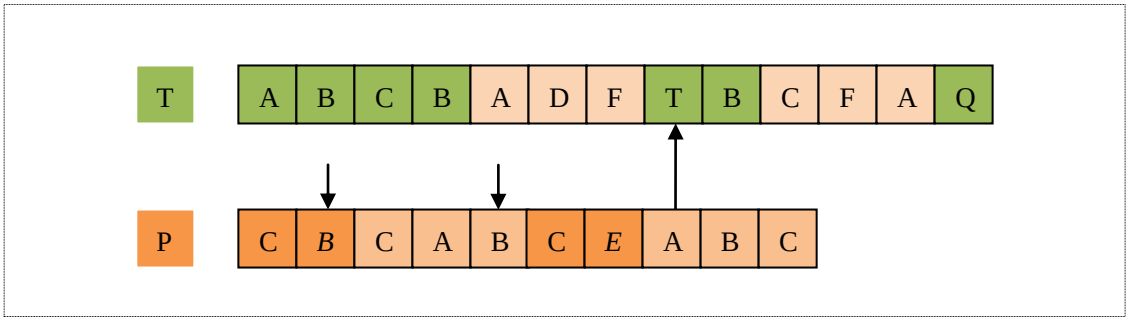
当然了，匹配过程中模板向右步长越大，匹配过程就越快，但前提是要能保证算法的正确性。

上图中，Good-Suffix 为 CAB，两类分析过程如下：

a) 根据 CAB 在 P 中未匹配字段中寻找 CAB 相同但 P 中 CAB 前一个字符不同的字段 T’，然后移动 P 使得 T’对齐 T 中的 CAB。
b) 如果不存在上述的 T’，那么就要搜索 P 中的最大前缀，并且这个前缀是 CAB 的最长后缀。

举例如下：

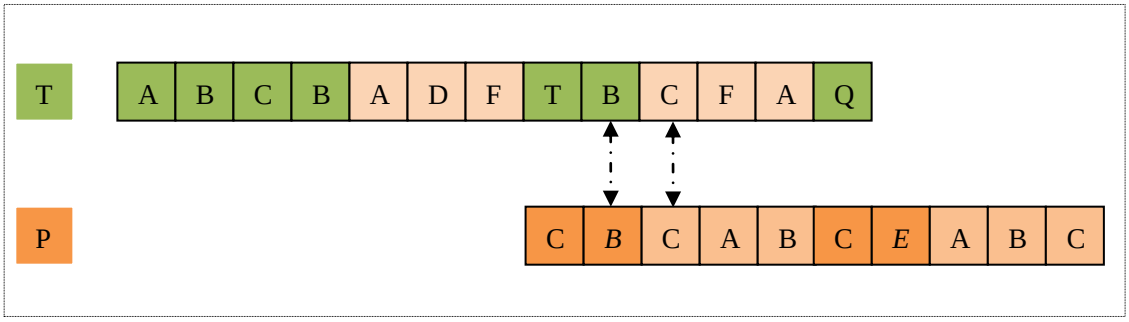
例一（说明情况 a）：



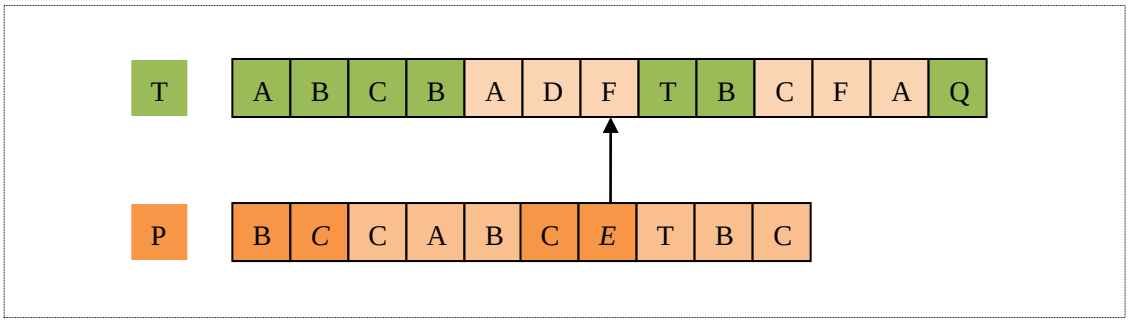
共同后缀为 TBC，寻找 P 中未匹配部分中是否出现过 BC。发现出现过两次，分别为 P 中的第 2 个位置和第 5 个位置。

但是第 5 个位置的 BC 前一个字符 A 跟匹配中的 BC 的前一个字符 A 相同，因此不符合要求。

所以移动过程只能是 2 位置。移动图示为：



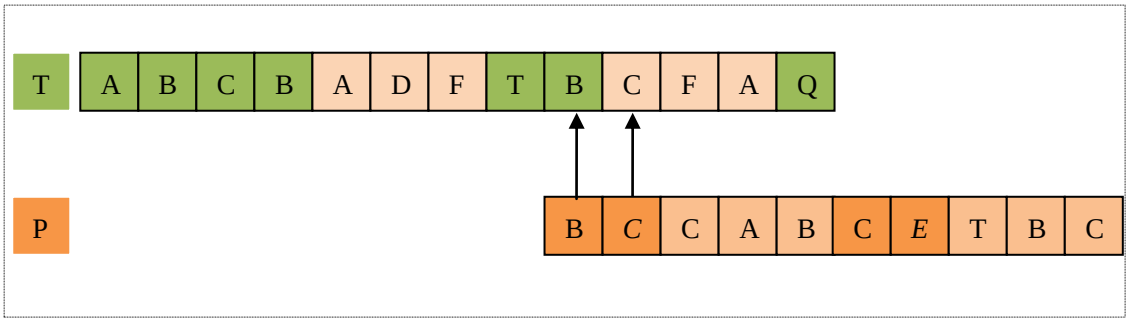
例二（说明情况 b）：



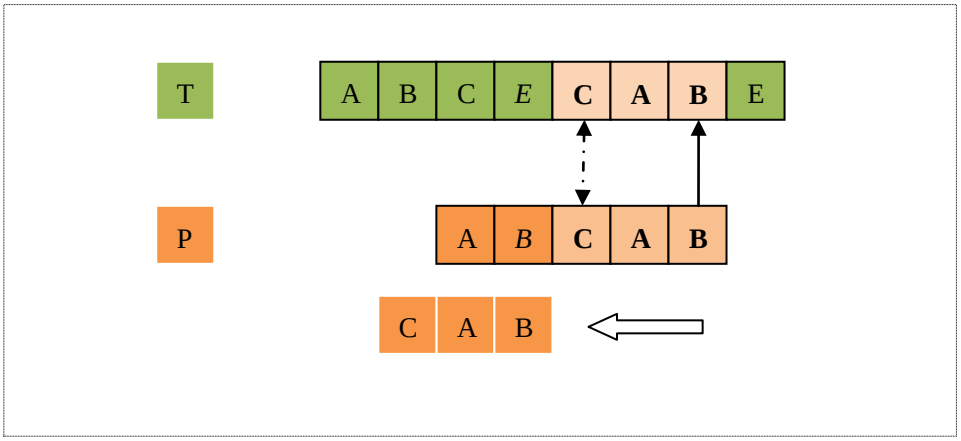
共同后缀为 TBC，寻找 P 中未匹配部分中是否出现过 TBC。发现未曾出现过。

那么我们就要找 P 的最长前缀同时又是 TBC 的最大后缀的情况。

发现只有 BC，那么 P 需要移动前缀 BC 至对齐 T 中的 BC。



经过上面两种情况的详细讲解，原文的 TP 匹配就要做下面的移动，图解如下：

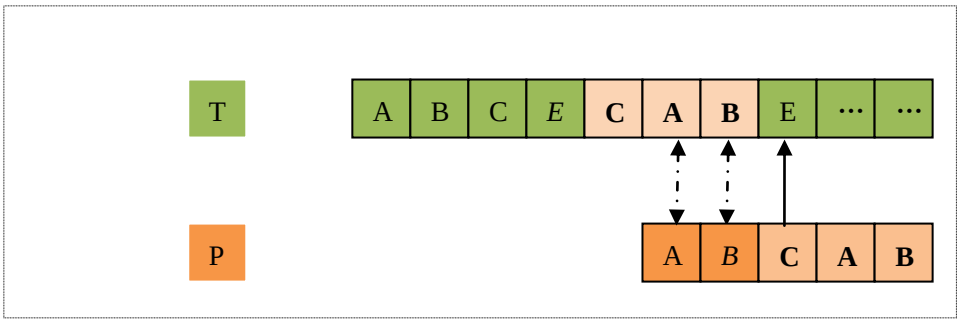


由于 CAB 在前面 P 中未曾出现过，只能进行第二种情况的最大前缀的匹配。

上图中对应的就是已经匹配的部分 CAB 字段在 P 中前方的最大重叠 AB。

看出来了吧，最大的移动就是让 P 中的其实部分 AB 跟 T 中已匹配的字段 CAB 的部分进行对齐。

移动后的结果如下：



自此，讲解完毕。

在 BM 算法匹配的过程中，取 Jump(x)与 Shift(x)中的较大者作为跳跃的距离。

BM 算法预处理时间复杂度为 O（m+s），空间复杂度为 O(s)，s 是与 P, T 相关的有限字符集长度，搜索阶段时间复杂度为 O(m*n)。

最好情况下的时间复杂度为 O(n/m)，最坏情况下时间复杂度为 O(m*n)。