**BM算法**

    KMP的匹配是从模式串的开头开始匹配的，而1977年，德克萨斯大学的Robert S. Boyer教授和J Strother Moore教授发明了一种新的字符串匹配算法：Boyer-Moore算法，简称BM算法。该算法从模式串的尾部开始匹配，且拥有在最坏情况下O(N)的时间复杂度。在实践中，比KMP算法的实际效能高。

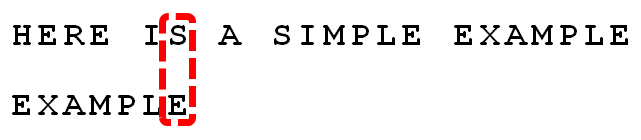
    BM算法定义了两个规则：

坏字符规则：当文本串中的某个字符跟模式串的某个字符不匹配时，我们称文本串中的这个失配字符为坏字符，此时模式串需要向右移动，移动的位数 = 坏字符在模式串中的位置 - 坏字符在模式串中最右出现的位置。此外，如果"坏字符"不包含在模式串之中，则最右出现位置为-1。

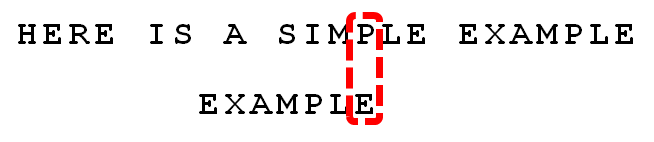
好后缀规则：当字符失配时，后移位数 = 好后缀在模式串中的位置 - 好后缀在模式串上一次出现的位置，且如果好后缀在模式串中没有再次出现，则为-1。

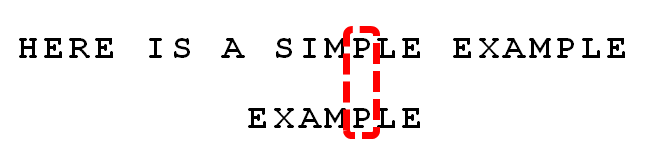
    下面举例说明BM算法。例如，给定文本串“HERE IS A SIMPLE EXAMPLE”，和模式串“EXAMPLE”，现要查找模式串是否在文本串中，如果存在，返回模式串在文本串中的位置。

    1. 首先，"文本串"与"模式串"头部对齐，从尾部开始比较。"S"与"E"不匹配。这时，"S"就被称为"坏字符"（bad character），即不匹配的字符，它对应着模式串的第6位。且"S"不包含在模式串"EXAMPLE"之中（相当于最右出现位置是-1），这意味着可以把模式串后移6-(-1)=7位，从而直接移到"S"的后一位。

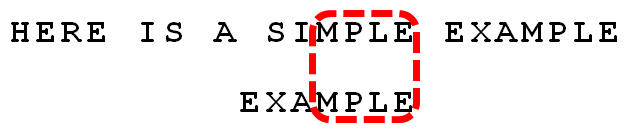


    2. 依然从尾部开始比较，发现"P"与"E"不匹配，所以"P"是"坏字符"。但是，"P"包含在模式串"EXAMPLE"之中。因为“P”这个“坏字符”对应着模式串的第6位（从0开始编号），且在模式串中的最右出现位置为4，所以，将模式串后移6-4=2位，两个"P"对齐。

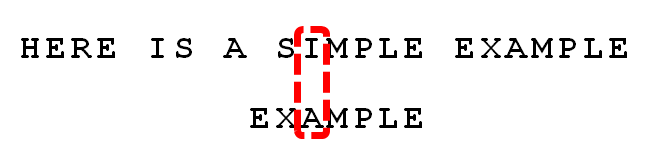


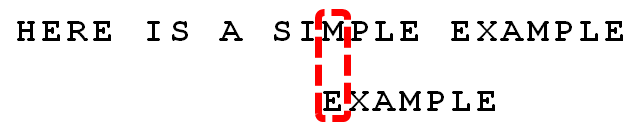


    3. 依次比较，得到 “MPLE”匹配，称为"好后缀"（good suffix），即所有尾部匹配的字符串。注意，"MPLE"、"PLE"、"LE"、"E"都是好后缀。

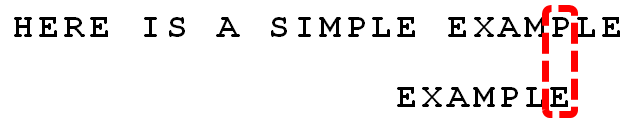


    4. 发现“I”与“A”不匹配：“I”是坏字符。如果是根据坏字符规则，此时模式串应该后移2-(-1)=3位。问题是，有没有更优的移法？

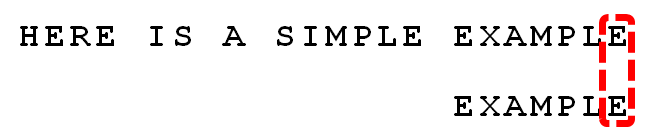




    5. 更优的移法是利用好后缀规则：当字符失配时，后移位数 = 好后缀在模式串中的位置 - 好后缀在模式串中上一次出现的位置，且如果好后缀在模式串中没有再次出现，则为-1。  
    所有的“好后缀”（MPLE、PLE、LE、E）之中，只有“E”在“EXAMPLE”的头部出现，所以后移6-0=6位。  
    可以看出，“坏字符规则”只能移3位，“好后缀规则”可以移6位。每次后移这两个规则之中的较大值。这两个规则的移动位数，只与模式串有关，与原文本串无关。



    6. 继续从尾部开始比较，“P”与“E”不匹配，因此“P”是“坏字符”，根据“坏字符规则”，后移 6 - 4 = 2位。因为是最后一位就失配，尚未获得好后缀。



    由上可知，BM算法不仅效率高，而且构思巧妙，容易理解。