

作者：阮一峰

上一篇文章，我介绍了 KMP 算法。

但是，它并不是效率最高的算法，实际采用并不多。各种文本编辑器的"查找"功能（Ctrl+F），大多采用 Boyer-Moore 算法。

Boyer-Moore 算法不仅效率高，而且构思巧妙，容易理解。1977 年，德克萨斯大学的 Robert S. Boyer 教授和 J Strother Moore 教授发明了这种算法。

下面，我根据 Moore 教授自己的例子来解释这种算法。


1.

字符串	HERE IS A SIMPLE EXAMPLE
搜索词	EXAMPLE

假定字符串为"HERE IS A SIMPLE EXAMPLE"，搜索词为"EXAMPLE"。

2.

HERE IS A SIMPLE EXAMPLE
EXAMPLE



首先，"字符串"与"搜索词"头部对齐，从尾部开始比较。

这是一个很聪明的想法，因为如果尾部字符不匹配，那么只要一次比较，就可以知道前 7 个字符（整体上）肯定不是要找的结果。

我们看到，"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。我们还发现，"S"不包含在搜索词"EXAMPLE"之中，这意味着可以把搜索词直接移到"S"的后一位。

3.

HERE IS A SIMPLE EXAMPLE
EXAMPLE



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"包含在搜索词"EXAMPLE"之中。所以，将搜索词后移两位，两个"P"对齐。

4.

HERE IS A SIMPLE EXAMPLE
EXAMPLE



我们由此总结出"坏字符规则":

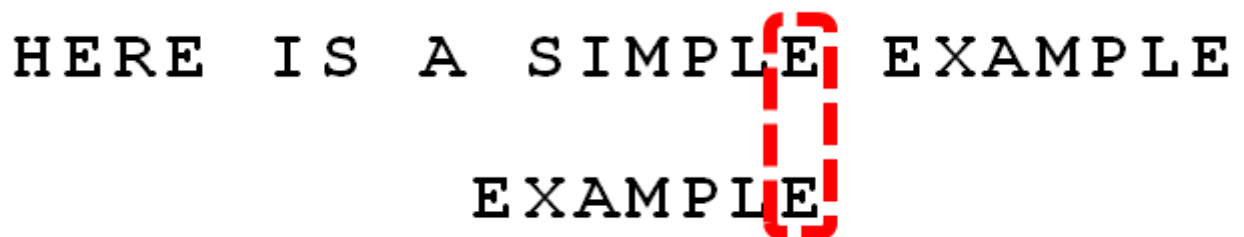
后移位数 = 坏字符的位置 - 搜索词中的上一次出现位置

如果"坏字符"不包含在搜索词之中, 则上一次出现位置为 -1。

以"P"为例, 它作为"坏字符", 出现在搜索词的第 6 位 (从 0 开始编号), 在搜索词中的上一次出现位置为 4, 所以后移 $6 - 4 = 2$ 位。再以前面第二步的"S"为例, 它出现在第 6 位, 上一次出现位置是 -1 (即未出现), 则整个搜索词后移 $6 - (-1) = 7$ 位。

5.


HERE IS A SIMPLE EXAMPLE
EXAMPLE



依然从尾部开始比较，"E"与"E"匹配。

6.


HERE IS A SIMPLE EXAMPLE
EXAMPLE



比较前面一位，"LE"与"LE"匹配。

7.

HERE IS A SIMPLE EXAMPLE
EXAMPLE



比较前面一位，"PLE"与"PLE"匹配。

8.

HERE IS A SIMPLE EXAMPLE
EXAMPLE



比较前面一位，"MPLE"与"MPLE"匹配。我们把这种情况称为"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。

9.



HERE IS A SIMPLE EXAMPLE
EXAMPLE

比较前一位，发现"I"与"A"不匹配。所以，"I"是"坏字符"。

10.



HERE IS A SIMPLE EXAMPLE
EXAMPLE

根据"坏字符规则"，此时搜索词应该后移 $2 - (-1) = 3$ 位。

问题是，此时有没有更好的移法？

11.

HERE IS A SIMPLE EXAMPLE
EXAMPLE

我们知道，此时存在"好后缀"。所以，可以采用"好后缀规则"：

后移位数 = 好后缀的位置 - 搜索词中的上一次出现位置

举例来说，如果字符串"ABCDAB"的后一个"AB"是"好后缀"。那么它的位置是 5（从 0 开始计算，取最后的"B"的值），在"搜索词中的上一次出现位置"是 1（第一个"B"的位置），所以后移 $5 - 1 = 4$ 位，前一个"AB"移到后一个"AB"的位置。

再举一个例子，如果字符串"ABCDEF"的"EF"是好后缀，则"EF"的位置是 5，上一次出现的位置是 -1（即未出现），所以后移 $5 - (-1) = 6$ 位，即整个字符串移到"F"的后一位。

这个规则有三个注意点：

(1) "好后缀"的位置以最后一个字符为准。假定"ABCDEF"的"EF"是好后缀，则它的位置以"F"为准，即 5（从 0 开始计算）。

(2) 如果"好后缀"在搜索词中只出现一次，则它的上一次出现位置为 -1。 比如，"EF"在"ABCDEF"之中只出现一次，则它的上一次出现位置为-1（即未出现）。

(3) 如果"好后缀"有多个，则除了最长的那个"好后缀"，其他"好后缀"的上一次出现位置必须在头部。 比如，假定"BABCDAB"的"好后缀"是"DAB"、"AB"、"B"，请问这时"好后缀"的上一次出现位置是什么？回答是，此时采用的好后缀是"B"，它的上一次出现位置是头部，即第 0 位。这个规则也可以这样表达：如果最长的那个"好后缀"只出现一次，则可以把搜索词改写成如下形式进行位置计算 "(DA)BABCDAB"，即虚拟加入最前面的"DA"。

回到上文的这个例子。此时，所有的"好后缀"（MPLE、PLE、LE、E）之中，只有"E"在"EXAMPLE"还出现在头部，所以后移 $6 - 0 = 6$ 位。

12.

HERE IS A SIMPLE EXAMPLE
EXAMPLE

可以看到，"坏字符规则"只能移 3 位，"好后缀规则"可以移 6 位。所以，Boyer-Moore 算法的基本思想是，每次后移这两个规则之中的较大值。

更巧妙的是，这两个规则的移动位数，只与搜索词有关，与原字符串无关。因此，可以预先计算生成《坏字符规则表》和《好后缀规则表》。使用时，只要查表比较一下就可以了。

13.

HERE IS A SIMPLE EXAMPLE
EXAMPLE

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。

14.

HERE IS A SIMPLE EXAMPLE
EXAMPLE

从尾部开始逐位比较，发现全部匹配，于是搜索结束。如果还要继续查找（即找出全部匹配），则根据"好后缀规则"，后移 $6 - 0 = 6$ 位，即头部的"E"到尾部的"E"的位置。

（完）

留言（44 条）

MGhostSoft 说：

本科的时候看算法课本就这里不明白，现在终于弄懂了。

2013 年 5 月
3 日
13:58 | [档](#)
[案](#) | [引用](#)

zhiyelee 说：

太赞了，看了上一篇意犹未尽。

2013 年 5 月
3 日
14:06 | [档](#)
[案](#) | [引用](#)

Bingfei 说：

通俗易懂的解释，如果教科书能按照这样编写，计算机会更有趣一些。

2013 年 5 月
3 日
14:18 | [档](#)
[案](#) | [引用](#)

Will Shen 说：

受教了，阮兄每篇都很值得我輩細讀。

2013 年 5 月
3 日
15:08 | [档](#)
[案](#) | [引用](#)

t.k. 说:

我也受教了，不过我想知道下面两处的来源：

- 1.文本编辑器的"查找"功能大多采用 Boyer-Moore 算法。
- 2.Boyer-Moore 算法效率高于 KMP。（KMP 的时间复杂度是 $O(n + k)$ ，但是 Boyer-Moore 是多少呢？）

2013 年 5 月
3 日
15:40 | [档](#)
[案](#) | [引用](#)

bluesleaf 说:

阮兄，

后移位数 = 好后缀的位置 - 搜索词中的上一次出现位置
计算时，位置的取值以"好后缀"的最后一个字符为准。

这里 "好后缀"的最后一个字符 不就只会是搜索词的最后一个字符吗？也就是说公式里的 好后缀的位置 就是 搜索词的长度-1（因为从 0 开始编号）吧？

2013 年 5 月
3 日
16:24 | [档](#)
[案](#) | [引用](#)

funnyfan 说:

“好后缀的位置”和“搜索词中的上一次出现位置”这两个我总是会弄混。。。

2013 年 5 月
3 日
16:38 | [档](#)
[案](#) | [引用](#)

HCocoa 说:

引用 funnyfan 的发言:

“好后缀的位置”和“搜索词中的上一次出现位置”这两个我总是会弄混。。。

“搜索词中的上一次出现位置”这个博主没解释清楚

2013 年 5 月
3 日
16:45 | 档
案 | 引用

阮一峰 说:

引用 t. k. 的发言:

我想知道下面两处的来源:

.....

可以参见[这个链接](#)。

2013 年 5 月
3 日
18:49 | 档
案 | 引用

阮一峰 说:

引用 HCocoa 的发言:

“搜索词中的上一次出现位置”这个博主没解释清楚

我加了两个例子，是不是好一点了？

2013 年 5 月
3 日
19:02 | 档
案 | 引用

一个程序员 说:

学习了，想到一个问题，如果从后面往前找这个算法该如何实现呢，规则还一样吗？

2013 年 5 月
3 日
20:31 | 档
案 | 引用

MaskRay 说:

有效 Boyer-Moore algorithm 實現的難點在于好後綴的計算，樸素的方法時間複雜度是 $O(m^2)$ 的， $O(m)$ 的計算還需要用到 Manacher algorithm。

[2013年5月3日](#)
[21:10](#) | [档案](#) | [引用](#)

biaobiaoqi 说:

另外，请教阮老师，贵博客的留言系统是自己做的定制开发还是用了哪个第三方的系统？感觉交互上比市面上常见的友言要好。多谢指导；)

[2013年5月3日](#)
[21:12](#) | [档案](#) | [引用](#)

ChanneW 说:

确实相当的巧妙！

[2013年5月3日](#)
[21:28](#) | [档案](#) | [引用](#)

xx 说:

此时，所有的"好后缀"（MPLE、PLE、LE、E）之中，只有"E"在"EXAMPLE"之中出现两次，所以后移 $6 - 0 = 6$ 位。

所以个啥？

觉得英文版更清楚

[2013年5月3日](#)
[21:32](#) | [档案](#) | [引用](#)

雪月秋水 说:

要是在来一篇字符串模糊匹配算法就好了~

2013年5月
3日
21:44 | 档
案 | 引用

sss 说:

看得感觉还不是特别懂.....

说一下 "3."这张图的到"4."这一步计算我的理解

以"3."为例

坏位置为重"example"对应的长字符串从后面开始查找,与上面位置对应不匹配的字第一次出现的位置.

比如图"3."以为例,经过前面的位移后,"example"的最后一位"e"对应长字符串的"p";

因为 $p \neq e$;所以:

坏位置 = "example"中"e"的位置(也就是 6)

但是 p 在"example",所以:

搜索词中的上一次出现位置 = p 在"example"中的位置(也就是 4)

不知道我的理解是否正确?

2013年5月
3日
21:59 | 档
案 | 引用

hejianchao 说:

同求: 字符串模糊匹配算法

2013年5月
3日

[22:01](#) | [档案](#) | [引用](#)

Onyas 说:

看了楼主两篇文章受益匪浅，，之前看了好多关于 KMP 与 BM 的文章和书都是一知半解，楼主解释的很清晰，，不知啥时候写下 KMP 和 BM 的实现，，网上许多写的都乱七八糟的

[2013 年 5 月 3 日](#)

[22:49](#) | [档案](#) | [引用](#)

hancy 说:

阮老师写的真好。

看的过程中只有一点迷糊了下：所有的"好后缀"（MPLE、PLE、LE、E）之中，是先选最长的“好后缀”MPLE，去找它在"搜索词中的上一次出现位置"，没有才选短一点的 PLE，以此类推，直到最短的好后缀？是这样吧？

[2013 年 5 月 4 日](#)

[10:21](#) | [档案](#) | [引用](#)

阮一峰 说:

@hancy:

对，就是这样。

因为“好后缀规则”比较麻烦，有些实现就只部署“坏字符规则”。

[2013 年 5 月 4 日](#)

[12:15](#) | [档案](#) | [引用](#)

adam 说:

这个算法设计比较精巧。

看阮兄的一系列文章，收益匪浅，尤其是《TF-IDF 与余弦相似性的应

用》系列，让大家明白其实算法和实际应用结合后，并不是那么的令人生畏。

[2013年5月4日](#)
[15:31 | 档案 | 引用](#)

花生 说:

关于坏字符的上一次位置，可能让人不理解。

上一次位置：如果有多个坏字符，指搜索词的最后一个坏字符的位置。这样是不是好点？

[2013年5月5日](#)
[15:12 | 档案 | 引用](#)

花生 说:

关于好后缀，如果有多个好后缀，则选择位于搜索词的头部，即第 0 位的那个，否则就是 -1，我是这样理解的。

“除了最长的那个”好后缀”，这句有点不理解。

[2013年5月5日](#)
[15:26 | 档案 | 引用](#)

Randoms 说:

这个怎么和 **linuxeden** 开源社区新浪微博上的文章是一样的。

[2013年5月5日](#)
[17:53 | 档案 | 引用](#)

shunner 说:

也是关于“上一次出现位置”。我想可不可以当作“最后一次出现位置”？以“坏字符”为例，就是在搜索词里从后往前找到的第一个“坏字符”的位置。这在搜索词中包含多个“坏字符”时，能够保证不会后移太多。

2013 年 5 月
5 日
18:39 | 档
案 | 引用

navono 说:

坏字符规则表和《好后缀规则表怎么生成？

它们都要根据字符在搜索词中出现的位置，怎么说和原字符（被搜索词？）无关？

2013 年 5 月
7 日
08:57 | 档
案 | 引用

navono 说:

引用 navono 的发言:

坏字符规则表和《好后缀规则表怎么生成？

它们都要根据字符在搜索词中出现的位置，怎么说和原字符（被搜索词？）无关？

明白了。

2013 年 5 月
7 日
09:54 | 档
案 | 引用

freeboy1015 说:

坏字符规则中如何算出后移的位数呢？我在开源中国博客

（http://www.oschina.net/question/12_23429）中看到了计算坏字符移动数组 BmBc 的方法：

"BmBc 数组的定义，分两种情况。

1、 字符在模式串中有出现。如下图，BmBc['k']表示字符 k 在模式串中最后一次出现的位置，距离模式串串尾的长度。

我对这个也有疑惑：情况 1 中为什么 BmBc['k']是表示字符 k 在模式串中最后一次出现的位置，距离模式串串尾的长度。如果有多个 k 出现呢？

假设:

T: abcdbeccg

P: abcdbecg

现在 c 和 e 不匹配, 按照你的上面的方法计算的 $BmBc['c']$ 等于 1, 显然不对啊!

2013 年 5 月

7 日

17:20 | 档

案 | 引用

freeboy1015 说:

引用花生的发言:

关于坏字符的上一次位置, 可能让人不理解。

上一次位置: 如果有多个坏字符, 指搜索词的最后一个坏字符的位置。这样是不是好点?

我觉得定义靠谱点, 开源中国这篇将 BM 的代码你看了没, 我觉得计算坏字符的上一次位置数组的代码有问题 (如果这个字符重复多次的话, 它取的是最靠右的那个位置的)。

2013 年 5 月

7 日

17:26 | 档

案 | 引用

楚轩 说:

引用 t.k. 的发言:

Boyer-Moore 算法效率高于 KMP。(KMP 的时间复杂度是 $O(n + k)$, 但是 Boyer-Moore 是多少呢?)

效率是 $o(N/M)$

2013 年 5 月

9 日

08:36 | 档

案 | 引用

sea 说:

这个 **bm** 算法的好后缀部分和 **kmp** 算法的前缀函数一模一样吧？
猜测：**bm** 算法是在 **kmp** 算法基础上发展的，而且效率略高。

[2013 年 5 月
16 日
13:01 | 档
案 | 引用](#)

jihite 说：

阮老师 太厉害了，新的偶像诞生了！

[2013 年 8 月
29 日
10:33 | 档
案 | 引用](#)

francis 说：

有个地方想的不是很明白

12 里面说，“**Boyer-Moore** 算法的基本思想是，每次后移这两个规则之中的较大值。”

您能举个例子说明一下“坏字符规则”移位更大的情况吗？

谢谢了

[2013 年 9 月
23 日
13:09 | 档
案 | 引用](#)

jimshao 说：

BM 算法不是最高效的，还可以提升，关键是第二步，如果是坏字符，**BM** 算法是从下一个字母开始，如果是个好后缀，则要做适当的平移。实际无论是否是好后缀还是坏后缀，下一个字符都得参与下一次匹配，所以可以从下一个字符开始匹配，本例中下一个字符是个空格，显然是个坏字符，则第二次从字符 **A** 开始，最后一位不匹配，再看下一个字母 **E**，则做平移，以此类推。显然会比 **BM** 算法更高效。

[2013 年 10
月 4 日
19:57 | 档
案 | 引用](#)

kangzj 说:

得把“好后缀”和“上一次出现位置”的定义说明白吧，阮大

2013 年 10
月 8 日
17:12 | 档
案 | 引用

laigus 说:

引用 freeboy1015 的发言:

坏字符规则中如何算出后移的位数呢？我在开源中国博客

(http://www.oschina.net/question/12_23429) 中看到了计算坏字符移动数组 BmBc 的方法:

"BmBc 数组的定义，分两种情况。

1、 字符在模式串中有出现。如下图，BmBc['k']表示字符 k 在模式串中最后一次出现的位置，距离模式串末尾的长度。

我对这个也有疑惑：情况 1 中为什么 BmBc['k']是表示字符 k 在模式串中最后一次出现的位置，距离模式串末尾的长度。如果有多个 k 出现呢？

假设:

T: abcd**b**ccg

P: abcd**b**ecg

现在 c 和 e 不匹配，按照你的上面的方法计算的 BmBc['c']等于 1，显然不对啊！

我想知道坏字符规则怎么预处理

我还看过是 bmbc['k']取离结尾最长的，也就是第一个出现的

但不管哪种都不符合“上一次出现的位置”

也就是说 对于一个字符'k'，bmbc['k']与它不匹配的位置也有关，那么一个一维数组就能解决这个问题么？

2013 年 10
月 9 日
21:58 | 档
案 | 引用

CK 说:

好后缀是不是有些问题。按照文中说法，好后缀的位置就是：
最后一个字符的位置-最后一个位置的字符在匹配字符中出现的位置。
比如匹配字符是 `errtert`,它的好后缀无论怎么样都是： $6-3=3$ 。

2013 年 10 月 22 日
23:09 | [档案](#) | [引用](#)

cumirror 说:

讲得真好，将整个算法的匹配思路讲得淋漓尽致，谢谢~

2013 年 11 月 5 日
10:21 | [档案](#) | [引用](#)

KG 说:

认真看了两遍，才算是领悟到了要领。楼主解释清晰透彻！

2013 年 12 月 27 日
15:22 | [档案](#) | [引用](#)

宋小北 说:

讲的很赞哦，最常用的东西，还有这么多奥妙

2014 年 5 月 30 日
16:31 | [档案](#) | [引用](#)

purh 说:

很赞，不知楼主能否写一篇 `sunday` 算法

2014 年 7 月 24 日
09:19 | [档案](#) | [引用](#)

TargetNull 说:

引用花生的发言:

关于坏字符的上一次位置，可能让人不理解。

上一次位置：如果有多个坏字符，指搜索词的最后一个坏字符的位置。这样是不是好点？

怎么会有多个坏字符呢，你说的是不是好字符？比如 **ABCCCCDAB**，那么好字符显然应该是 **AB** 而不是你说的最后一个 **B**，我觉得博主讲的在前面补全的方法很好。

2014 年 8 月
29 日
19:09 | 档
案 | 引用

yy57 说:

好后缀这个地方有点没看懂。

如果好后缀不是一个字母，那么好后缀肯定不止一个对吧。

比如举例说的 **MPLE**，好后缀有 **MPLE**，**PLE**，**LE**，**E**。

但是取哪个后缀进行计算的时候没说清楚，我感觉。

是先看，次长后缀是不是出现在词头，没有，再看次之的，以此类推么？？？？

如果，所有的好后缀都没有出现在词头，那怎么办？？？

求解释。。。这个地方没看太懂，不过还是谢谢了！

说的挺透彻的！

2014 年 9 月
21 日
21:48 | 档
案 | 引用