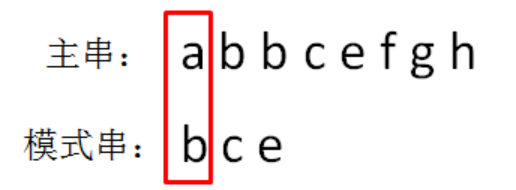
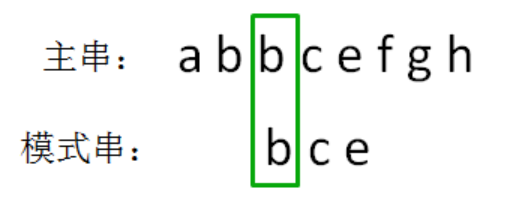
# **BF算法**

Brute Force（暴力算法）







极端情况：



假设主串的长度是m，模式串的长度是n，那么在这种极端情况下，BF算法的最坏时间复杂度是**O（mn）**。

# RK算法

Rabin-Karp, 是由算法的两位发明者Rabin & Karp的名字命名的

核心: 比较两个字符串的哈希值

生成hashcode的算法多种多样，比如：

1. **按位相加**

这是最简单的方法，我们可以把a当做1，b当做2，c当做3......然后把字符串的所有字符相加，相加结果就是它的hashcode。

bce =2 + 3 + 5 =10

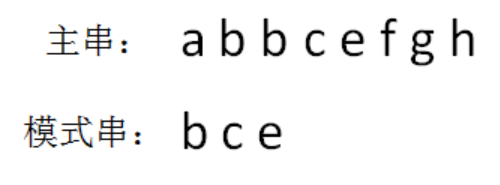
但是，这个算法虽然简单，却很可能产生hash冲突，比如bce、bec、cbe的hashcode是一样的。

1. **转换成26进制数**

既然字符串只包含26个小写字母，那么我们可以把每一个字符串当成一个26进制数来计算。

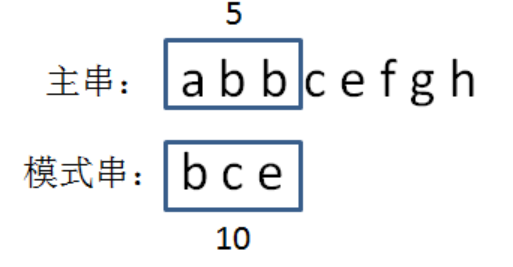
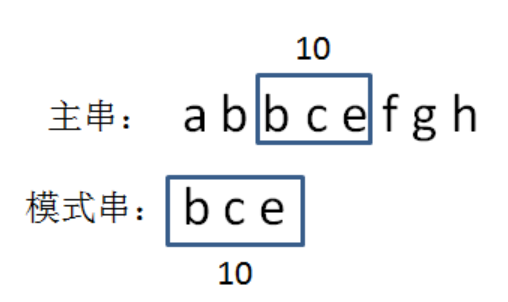
bce =2\*(26^2) +3\*26 +5 =1435

这样做的好处是大幅减少了hash冲突，缺点是计算量较大，而且有可能出现超出整型范围的情况，需要对计算结果进行取模。



1. 生成模式串的hashcode。

按位相加，bce = 10

1. 对主字符串等长字符进行hashcode计算  
   
2. 依次找到主串中hashcode=10的，然后依次比较字符是否相同  
   

时间复杂度：

假设主串的长度是m，模式串的长度是n，

最优为o(m)

最差每个等长都在hash冲突，又变成了o(mn)

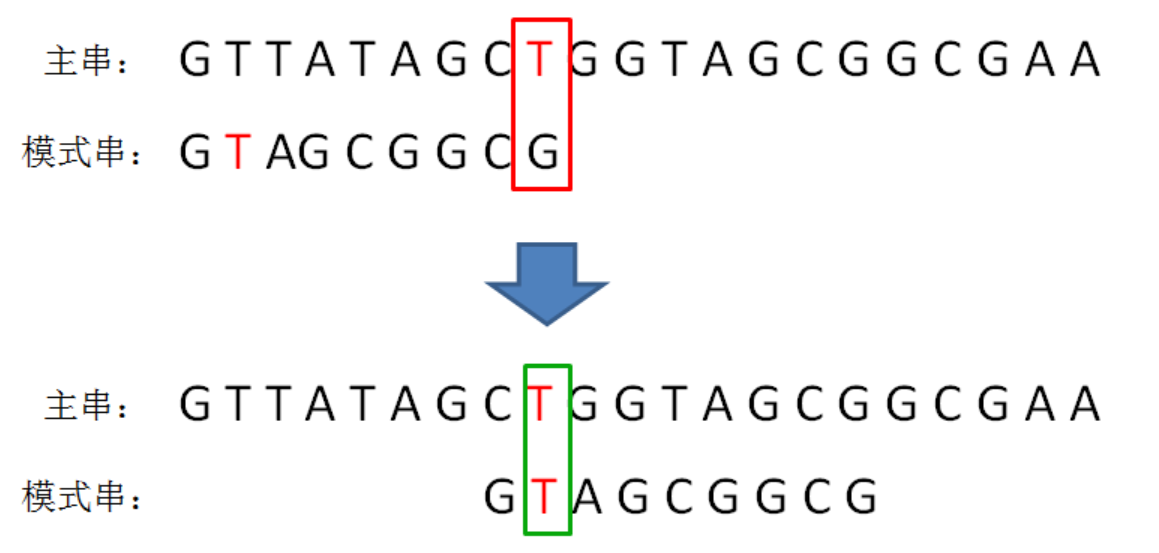
# BM算法

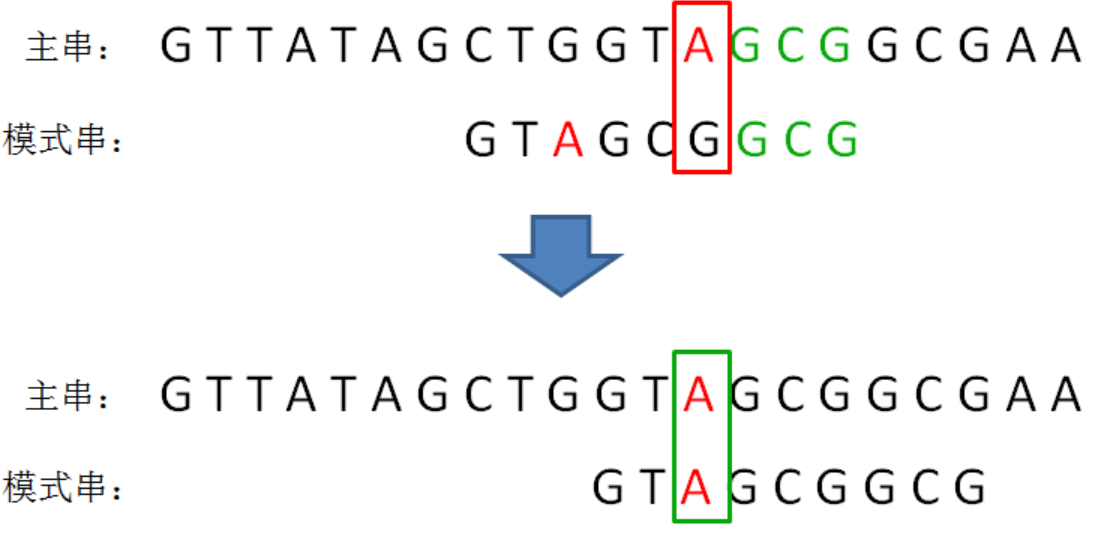
[BM算法 - 程序员小灰 - csdn](https://blog.csdn.net/bjweimengshu/article/details/104368394)

由两名计算机科学家Bob Boyer & JStrother Moore 发现

引入：坏字符规则 && 好后缀规则

## 坏字符规则



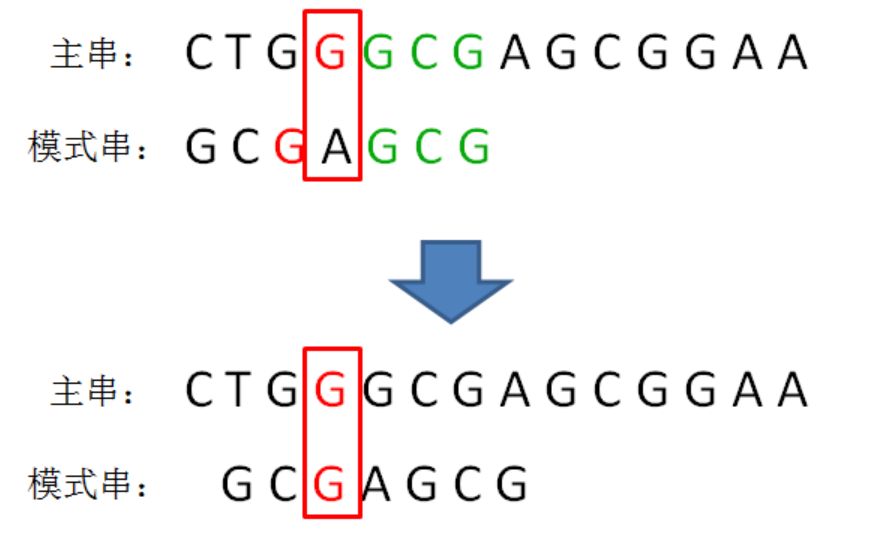


这个例子，三轮就搞定了，神人呀

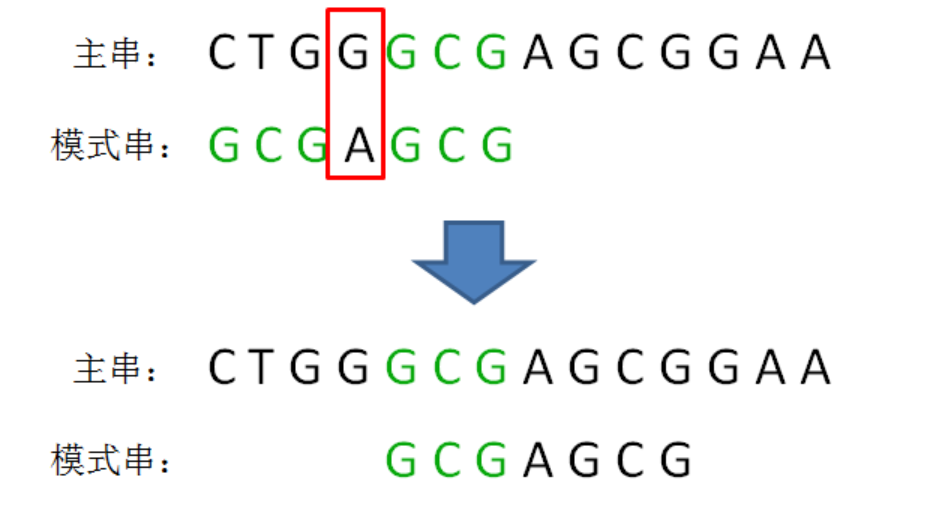
## 好后缀规则

代码比较难实现，仅记录思路

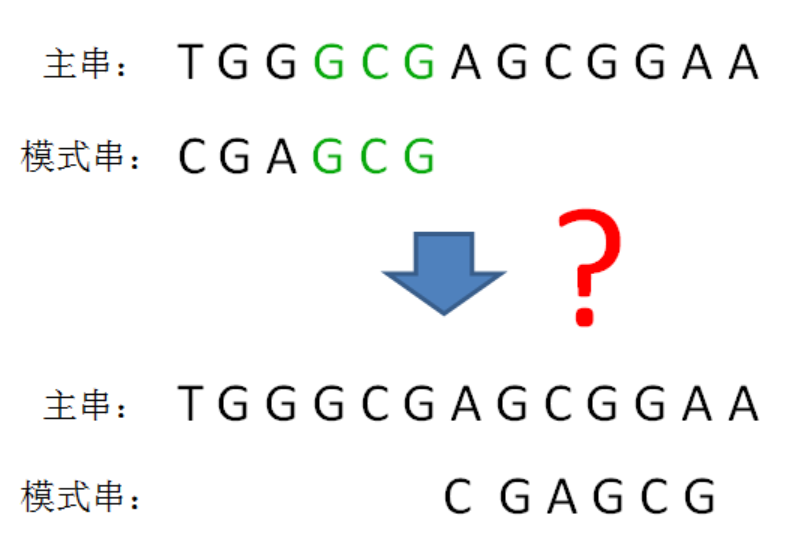
如果使用坏字符规则，只能挪动一位



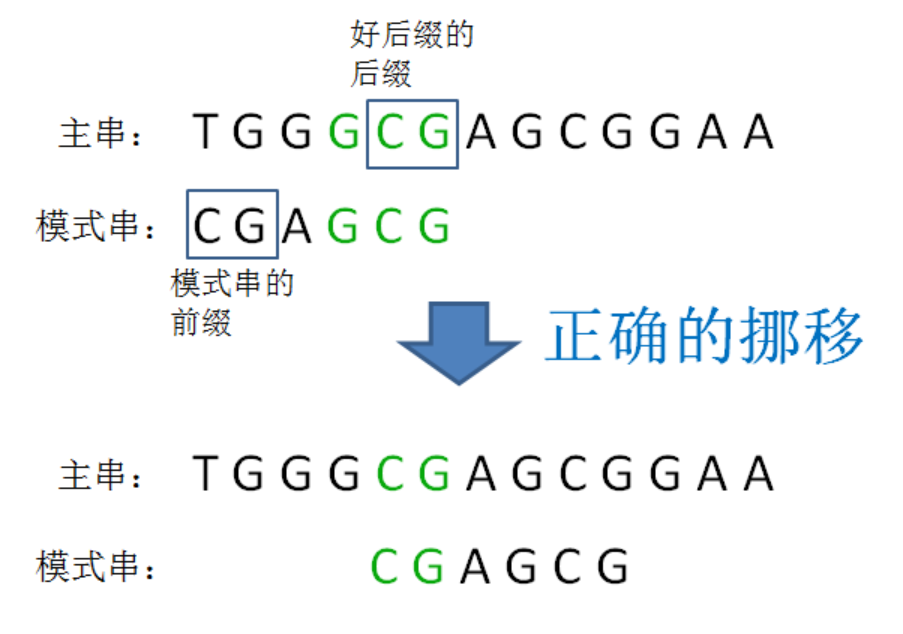
如果使用“好后缀”，则可以移动4位：



如果不存在“好后缀”，是否能直接挪动整个模式串长度呢？



不能！！！



所以结论：

1. 好后缀规则：
2. 通过坏字符串比较出了“好后缀“a
3. 从模式串的开头和a进行比对，看模式串开头能匹配多少，匹配的结果为b
4. A的其实位置C才是最后移动需要对齐的位置
5. 结合坏字符 & 好后缀，看哪一次移动距离长，那就选那一次了  
     
   算法复杂度：

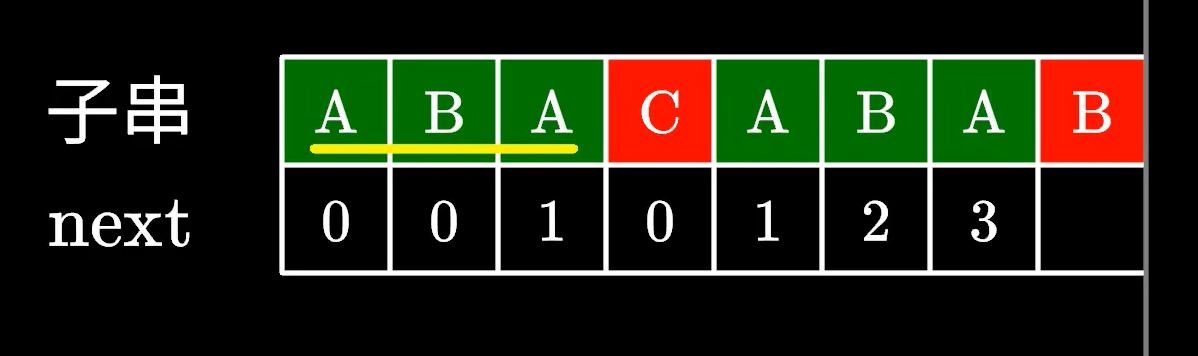
[BM算法](https://mp.weixin.qq.com/s/2RlyDBo-Ql-1Ofh8tMyikg)

# KMP算法

## 思想：

1. 和bm一样，开局还是寻找坏字符串；不过是从头往后找
2. 找到最长可匹配前缀 & 最长可匹配后缀  
   

## Next数组



Next数组长度 == 子串数组

Next的小标i表示已经和主串匹配了多少相同字符

Next[i] 表示长度为i的子串下，可以匹配多少最长匹配前缀/最后匹配后缀

Next[2] = 1: 字符串ABA，最长匹配长度为1

# References

[漫画：什么是字符串匹配算法？ - 程序员小灰 - csdn](https://blog.csdn.net/bjweimengshu/article/details/103966767)

[最浅显易懂的 KMP 算法讲解 - 奇乐编程学院 - bilibili](https://www.bilibili.com/video/BV1AY4y157yL/?spm_id_from=333.337.search-card.all.click&vd_source=10d355be19883e4a50b66949c50a67aa)