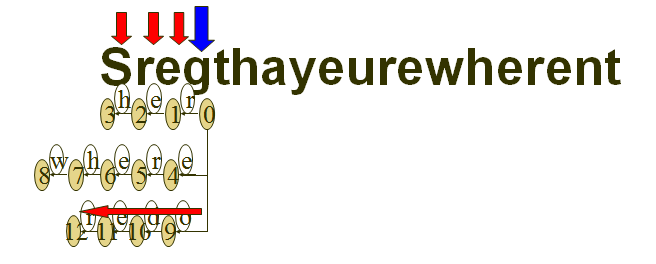
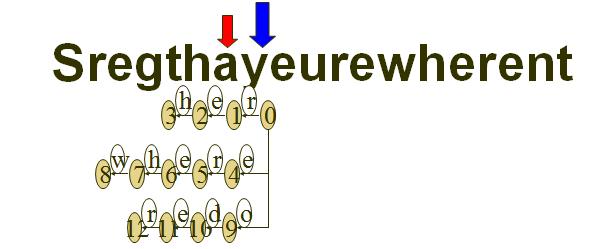
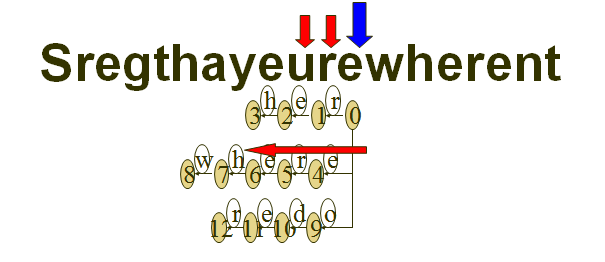
ACBM算法：

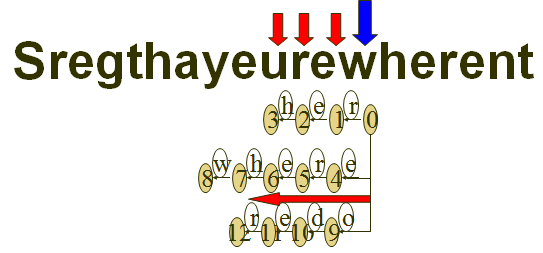
ACBM算法是在[AC自动机](http://blog.csdn.net/sealyao/article/details/4560427)的基础之上，引入了[BM算法](http://blog.csdn.net/sealyao/article/details/4568167)的多模扩展，实现的高效的多模匹配。和AC自动机不同的是，ACBM算法不需要扫描目标文本串中的每一个字符，可以利用本次匹配不成功的信息，跳过尽可能多的字符，实现高效匹配。

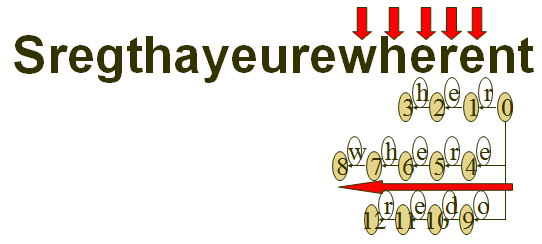
比如：{P} = {her，where，redo}，匹配过程如下：











可以看到在匹配步骤如下：

1、  选定一个匹配起始位置。

2、  使用AC树进行匹配，如果匹配失效，跳转步骤1；如果匹配成功，可根据应用需要跳转至步骤1或者退出。

ACBM算法的核心思想就是让每次匹配的起始位置跨度尽可能的大，以提高效率。

上例中，采用的是整体上正向匹配+反向的AC自动机；可以采用整体上反向的匹配+正向的AC自动机。

ACBM算法中的AC算法部分比AC自动机算法的实现要简单，不需要考虑失效函数的问题，也就是说ACBM算法中实现的AC算法部分是一棵树，而在AC自动机的实现是一个图。

ACBM算法中的BM算法的实现要比BM算法本身的实现要复杂一些，因为这是对BM算法的多模式一种扩展。

ACBM算法中的核心数据结构：

1、  MinLen，模式串集合中最短那个模式串的长度：比较失配时最多跳跃的字符个数不能超过Minlen。

2、  ACTree，由模式串集合构建出的状态树，构建方法和AC自动机的构建方法不同，采用反向构建自动机，而且不需要计算失效函数，比较简单。

3、  BCshift[256]：ACTree对应一个坏字符数组，当匹配失效时，查找该数组计算坏字符偏移量。

4、  GSshift：AC树的每一个节点对应一个好后缀偏移量。

ACBM算法的核心部分是计算BCshift和GSshift，可参考BM算法的实现。