大作业指导文档

一、 问题描述

用 O(m)时间复杂度找出一个长度为 m 的短字符串在一个长度为 n 的长字符串中的精确匹配，n>>m。

二、 数据

有三个长串还有对应的三组短串集合。

每个字符串都是一段蛋白质序列，只有四类字符

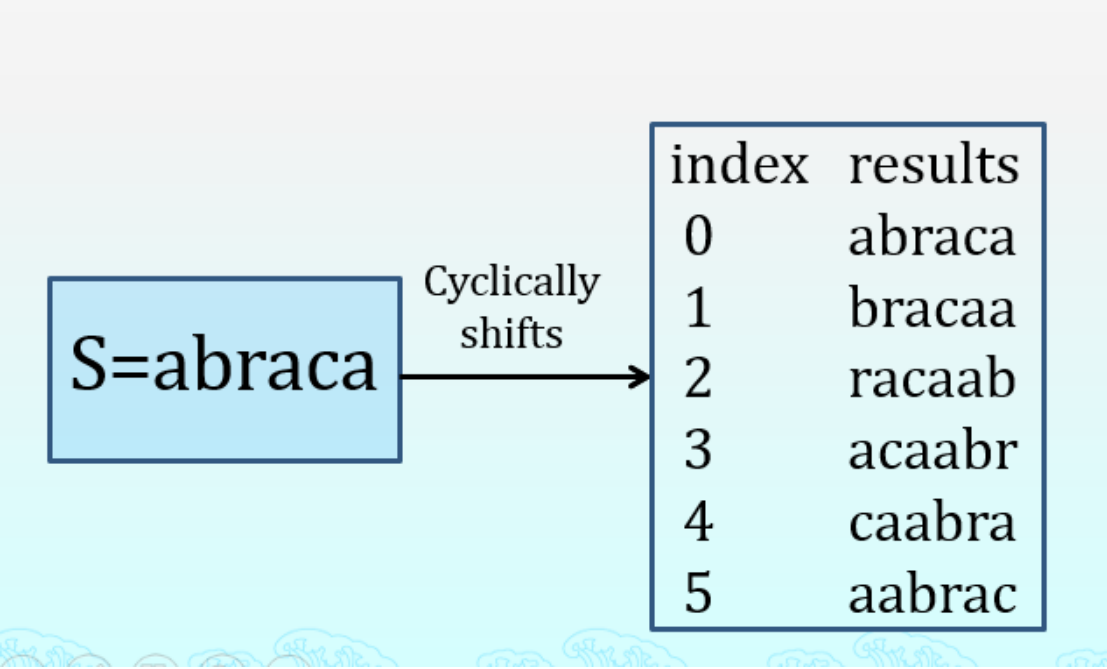
最长的一个长串大小有 900M，最后要求对最长的长串成功匹配即可。

三、 实现步骤

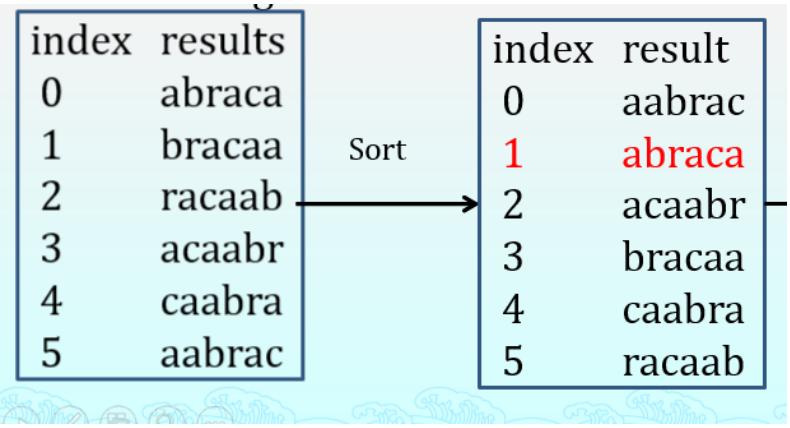
首先要实现 Burrows-Wheeler transform 算法并建立 FM 索引。

1. 建立 BWT 结构

首先对整个字符串每次右移一位生成新的字符串，所以对于长度为 n 的字符串会生成 n 个字符串。



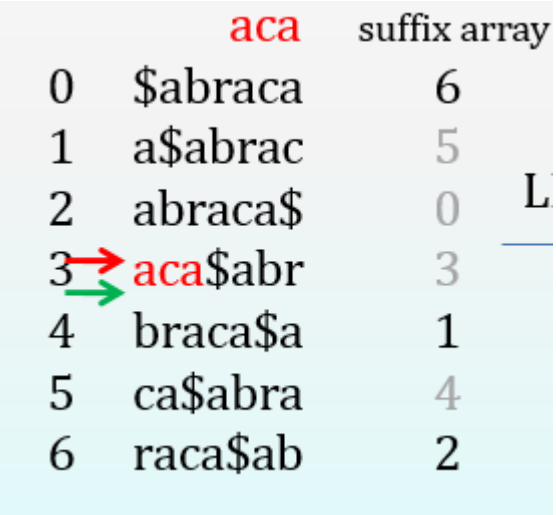
对这 n 个字符串按字典序排序，然后将**第一列和最后一列**取出。



需要注意的是若将字符串排序需要先创建 O（n^2）的空间，但因为长串大小为

900M，所以不可能生成整个字符串矩阵，需要分段生成 BWT，每段分别于短串做精确匹配，再将结果合并即可。这时有可能会出现段与段衔接部分的字符串匹配到短串的问题，所以需要段与段之间设置一定长度的重叠，重叠的长度只要大于等于短串长度即可解决该问题。

在做字典序排序的时候我们要同时建立 FM-index，也就是 suffix array。意思是我们排好序以后的第 k 个字符串是原字符串左移几位得到的，下图是个源字符串为 abraca$的例子，后缀数组 S[i]存的是排序后第 i 个字符串是源字符串左移的位数。有了这个数组我们就可以通过匹配排序后的结果位置来定位源字符串中的位置。比如我们在排序后的第 4 个字符串的前三个字符匹配到了“aca”，那我们就可以通过 suffix array，S[3] = 3 来知道这个 aca 是在源字符串的第 4 个位置开始的。



2. **通过 BWT 生成的第一列字符串与最后一列字符串计算辅助数据结构（C 数组和 OCC 数组）**

   C 数组的元素 C[X]表示的意思是在字典序排序中（即第一列的字符串），比字符 X 小的有多少个。C 数组可以通过遍历第一列字符串来建立。拿上图的例子来说 C[a] = 0, C[b] = 3, C[c] = 4, C[r] = 5

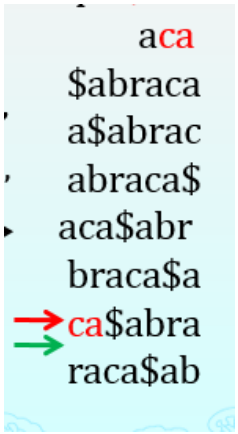
   OCC 数组是个二维数组，它的的元素 OCC[X, i]表示的意思是在最后一列字符串中，前 i 个字符里有几个字符 X。同样拿上图的例子来说以字符 a 为例，因为最后一列字符串的第一个字符是 c，所以 OCC[a, 1] = 0，第二个字符是 a 所以 OCC[a, 2] = 1，同理 OCC[a,3] = 1, OCC[a, 4] = 2, OCC[a, 5] = 3, OCC[a, 6] = 3

   因为我们最后的数据集中只有四种字符，所以通过最后一列字符串建立 OCC 数组也只需要 O(n)的时间。

3. **通过以上数据结构计算精确匹配**

   我们从这个第一列和最后一列字符串的构建方法上可以得到两个性质

第一个性质是第一列的第 i 个字符和最后一列的第 i 个字符在源字符串中是挨着的，并且最后一列的第 i 个字符在第一列第 i 个字符前面。这样我们就可以把短串从后往前匹配，先匹配目标串 S 的最后一个字符，然后拿到最后字符的在第一列字符串中的位置之后看这些位置的最后一列字符是否与 S 的倒数第二个字符匹配



如图里所示，需要匹配的小串是 aca，我们先取最后一个 a，在第一列中用 C 数组得到了 a 的位置范围为[1，3]，查看倒数第二个字符 c 是不是在最后一列字符串[1,3]的位置中存在，事实发现确实有一个 c，然后我们就定位到 ca 这个前缀在第一列中的位置 5，通过检查该位置在最后一个字符串中的字符为 a，发现确实与小串 aca 完全匹配，再定位到 aca 为前缀的位置 3 即可，最后通过 FM-index 建立的 suffix array 就可以找到 aca 在源字符串的位置了。

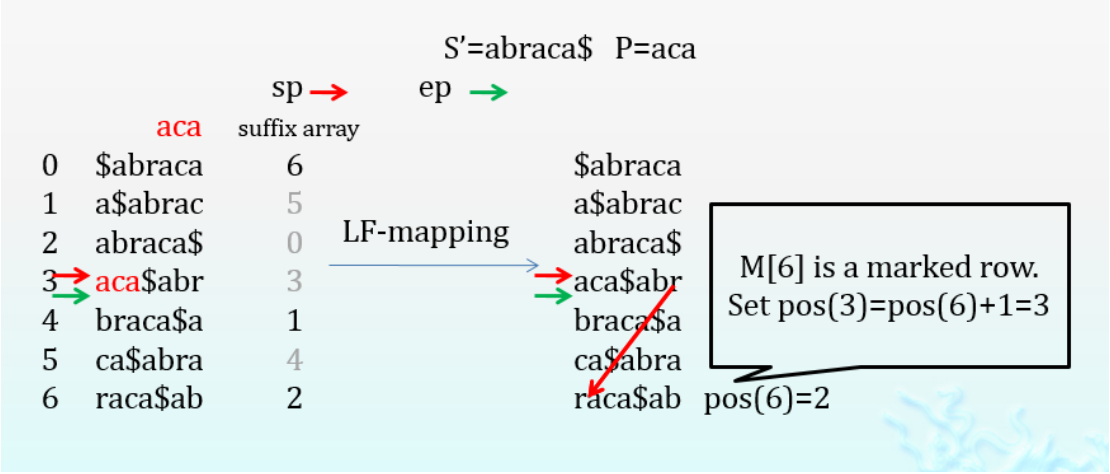
第二个性质是第一列字符串中的字符与最后一列字符串中的相同字符有顺序不变性，即第一列字符串中的第二个 a 与最后一列字符串中的第二个 a 在源字符串中的位置相同，即同一个 a。通过这个性质我们就可以把刚才的匹配方法中从[1,3]的区间变为 5 这个位置的逻辑实现。以上面的图为例，我们首先匹配到了位置 1 的字符串满足首位为 a，末位为 c，我们想定位以这个 ca 为前缀的字符串位置来做下一步的匹配，就只需先查 OCC[c, 1]来看最后一列字符串在位置 1 之前有几个 c，然后再通过 C[c]来定为第一列字符串中第一个 c 的位置，用 C[c] + OCC[c,1] = 5 + 0 = 5 即可定位到 ca 为前缀的字符串出现在 5 这个位置。

最后得到匹配的区间，再用后缀数组即可定位到匹配到的所有字符串在原串中的位置。

实现时需注意的地方：

1. 搜索串 searchXXX 中前几个字符与最后一个字符不是蛋白质序列，前面几个是记录长度的和制表符，最后一个字符是一个 OxFF，需要去掉后再处理。
2. 小的匹配串是每个大串的前 200,000 个字符。大家实现后只需要取其中几段测试正确性即可。
3. 索引要创建在本地，实现每次读索引文件查询。注意到 C、OCC、FM-index 后缀数组三个数据结构全部存储需要占用 O(4n + n)大小的 int 值，字符串大小约为 900M，字符存储需 1Byte，int 值存储需 4Byte，则总索引空间至少为 4*5*900M 约为 20G。空间占用太大，需要做一些部分存储的处理。
4. OCC 数组部分存储：OCC 数组是个二维数组，它的的元素 OCC[X, i]表示的意思是在最后一列字符串中，前 i 个字符里有几个字符 X。每当 i 隔 16 个或 32 个时存储一个 OCC[X,i]，比如存储 OCC[X,16]，OCC[X,32]，OCC[X,48]….。查询时先找到离得最近的存储点再计算真实值。

后缀数组部分存储：


S'=abraca$ P=aca
sp
ep
aca
suffix array
0 $abraca 6
1 a$abrac 5
2 abraca$ 0
3 aca$abr 3
4 braca$a 1
5 ca$abra 4
6 raca$ab 2
LF-mapping
$abraca
a$abrac
abraca$
aca$abr
braca$a
ca$abra
raca$ab
M[6] is a marked row.
Set pos(3)=pos(6)+1=3
pos(6)=2