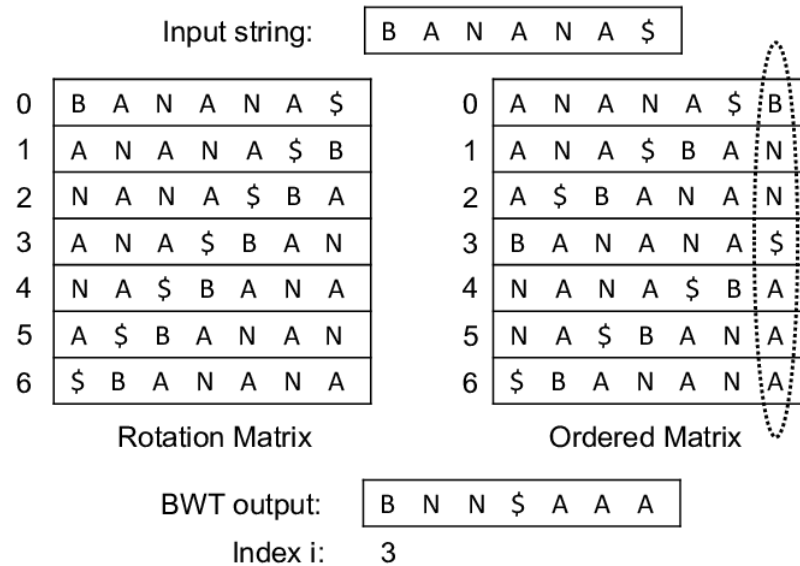


原始文档格式：

[<offset1>]<text1>[<offset2>]<text2>[<offset3>]<text3>... ..

BWT:



原始文档

[8]Computers in industry[9]Data compression[10]Integration[11]Big data indexing

经过 BWT 之后

agsan1[[1[[[[]]]]nngy1890ttDdr nnttrdineBzzzxstooiooiilcCiimmgpetrseuaanuasdper

RLB:

黄色表示连续重复出现 3 次以上的字符

用 1 开头的字节表示，剩余 7 bits 表示(重复次数-3)

1000 0000 表示重复 3 次

1000 0111 表示重复 10 次

如果 7 bits 不够表示重复次数，重复次数超过 2^7 次，用多个 1 开头的字节表示重复次数。

20,000 is represented as 19,997, in binary

1 0011100 0011101

and it will be divided into 3 bytes accordingly:

10011101 10011100 10000001.

RLB 之后

agsan1[[1[[[]]]nngy1890ttDdr nnttrdineB xstooiooiIcCiimmgpetrseuaanuasdper

在 RLB 文件中进行对于原始文档的搜索

搜索: "in"

返回: [8]Computers in industry

[11]Big data indexing

过程

重点：用 RLB 文件创建 C table 和 Occurrence table

C[i] 儲存 number of chars alphabetically less than i

比如 BANANA $C[A] = 0$ (A 最小) $C[B] = 3$ (3 个 A)

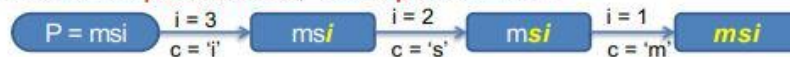
Occ[m,n] 儲存 number of occurrences of char m at the nth char of the file

比如 BNNAAA $\text{Occ}[A, 1] = 0$ $\text{Occ}[A, 4] = 1$ $\text{Occ}[A, 6] = 3$

用 Backward Search

BACKWARD-SEARCH

- Works in p iterations, from p down to 1



- Remember that the BWT matrix rows = sorted suffixes of T

- All suffixes prefixed by pattern P, occupy a continuous set of rows
- This set of rows has starting position **First** _____
- and ending position **Last** _____
- So, $(Last - First + 1)$ gives total pattern occurrences

- At the end of the i -th phase, **First** points to the first row prefixed by $P[i,p]$, and **Last** points to the last row prefixed by $P[i,p]$.

F	L ₂
# mississippi i	
i #mississippi p	
i ppi#mississ	
i sissippi#miss	
i sissippi#m	
m ississippi #	
p i#mississip	
p pi#mississ i	
s ippimississ	
s issippi#mi	
s issippi#miss i	
s ississippi#m i	

Algorithm backward_search($P[1, p]$)

(1) $i \leftarrow p$, $c \leftarrow P[p]$, **First** $\leftarrow C[c] + 1$, **Last** $\leftarrow C[c + 1]$;

```
(2) while ((First  $\leq$  Last) and ( $i \geq 2$ )) do
```

$$(3) \quad c \leftarrow P[i-1];$$

(4) $\text{First} \leftarrow C[c] + \text{Occ}(c, \text{First} - 1) + 1;$

(5) **Last** $\leftarrow C[c] + \text{Occ}(c, \text{Last});$

(6) $i \leftarrow i - 1;$

(7) if (Last < First) then return "no rows prefixed by $P[1, p]$ " else return (First, Last).

Backward search of the pattern 'si'

	F	Unknown	L	
	\$	mississipp	i	
Step 1:	{	\$mississip	p	Find first and last 's' and apply LF mapping
rows prefixed by 'i'		ppi\$missis	s	
		i ssippi\$mis	s	
		ssissippi\$	m	
		m ississippi	\$	
		p i\$mississi	p	
		p pi\$mississ	i	
Step 2:	{	ppi\$missi	s	
rows prefixed by 'si'		i ssippi\$mi	s	
		s sippi\$miss	i	
		s sissippi\$m	i	

得到 First 和 Last 两个数字

Last – First + 1 就是 “in” 总共在原始文件出现的次数

从 First 到 Last 进行 loop 操作

比如 29 – 33

29 放到 C table 得到 第一个原始 char 和这个 char 目前出现的次数 (第 n 个这个 char)

放到 Occ table, 得到第 n 个这个 char 在 L 里面的序数

又把序数放到 C table……

直到遇到 '[' 为止, 得到 in 的前半段原始字符, 后半段操作同理反过来

两边相加, 得到一条完整的 record:

[8]Computers in industry

30 放进 C table…

得到所有 5 个 records, 排序排重输出答案