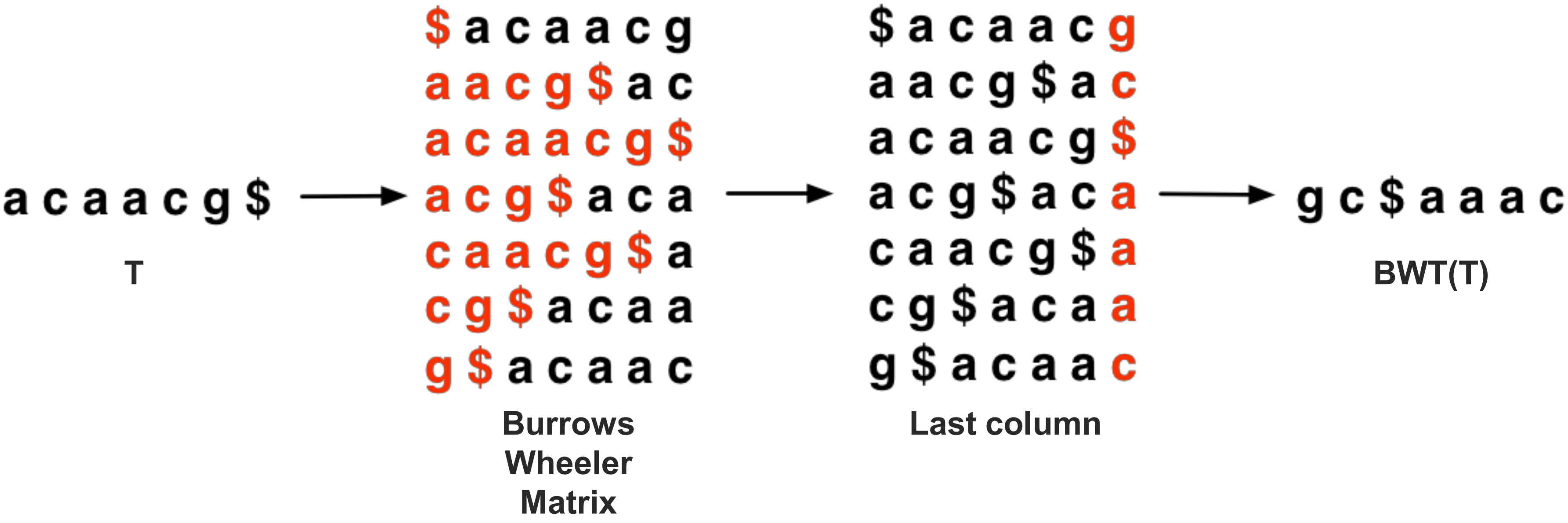
The Burrows-Wheeler Transform is a reversible representation with handy properties

- Sort all the possible rotations of original string



acaacg$ →

T

$acaacg
aacg$ac
acaacg$
acg$aca
caacg$a
cg$acaa
g$acaac

Burrows Wheeler Matrix

→

$acaacg
aacg$ac
acaacg$
acg$aca
caacg$a
cg$acaa
g$acaac

Last column

→ gc$aaac

BWT(T)

- Once BWT(T) is built, *all else shown here is discarded*
  - Matrix will be shown for illustration only

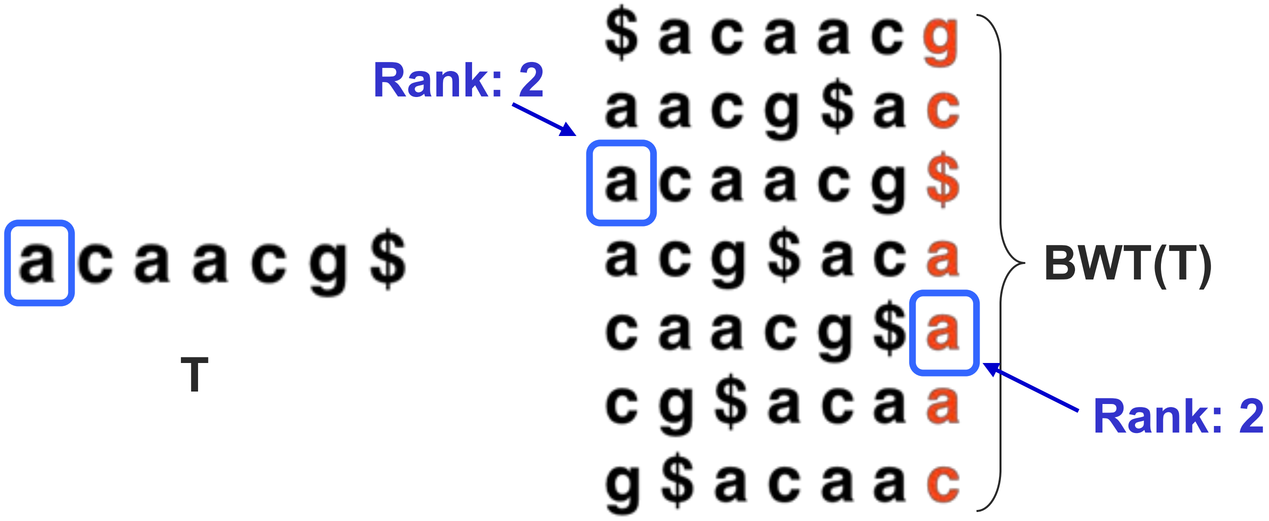
Burrows M, Wheeler DJ: **A block sorting lossless data compression algorithm**. *Digital Equipment Corporation, Palo Alto, CA* 1994, Technical Report 124; 1994

Courtesy of Ben Langmead. Used with permission.

http://www.cbcb.umd.edu/~langmead/NCBI_Nov2008.ppt

A text occurrence has the same rank in the first and last columns

- When we rotate left and sort, the first character retains its rank. Thus the same text occurrence of a character has the same rank in the **L**ast and **F**irst columns.



```
acaacg$

T
```

Rank: 2

```
$acaacg
aacg$ac
acaacg$
acg$aca
caacg$a
cg$acaa
g$acaac
```

BWT(T)

Rank: 2

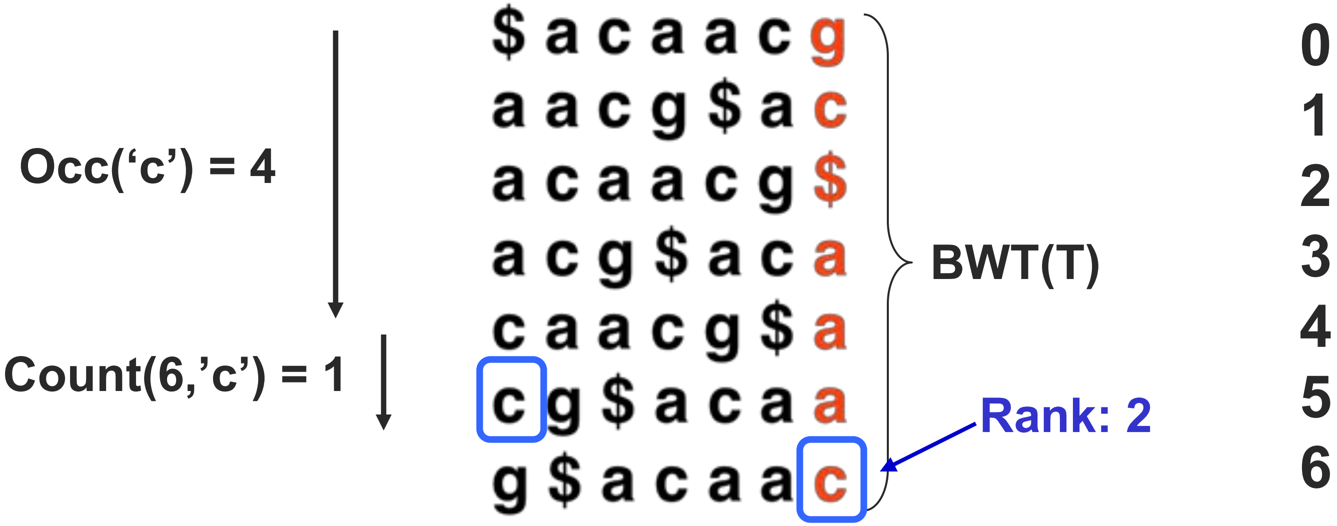
Burrows Wheeler Matrix

Courtesy of Ben Langmead. Used with permission.

http://www.cbcb.umd.edu/~langmead/NCBI_Nov2008.ppt

The Last to First (LF) function matches character and rank

LF(6, ‘c’) = Occ(‘c’) + Count(6,’c’) = 5



$ a c a a c g
a a c g $ a c
a c a a c g $
a c g $ a c a
c a a c g $ a
c g $ a c a a
g $ a c a a c

Occ(qc) – Number of characters lexically smaller than qc in BWT(T)

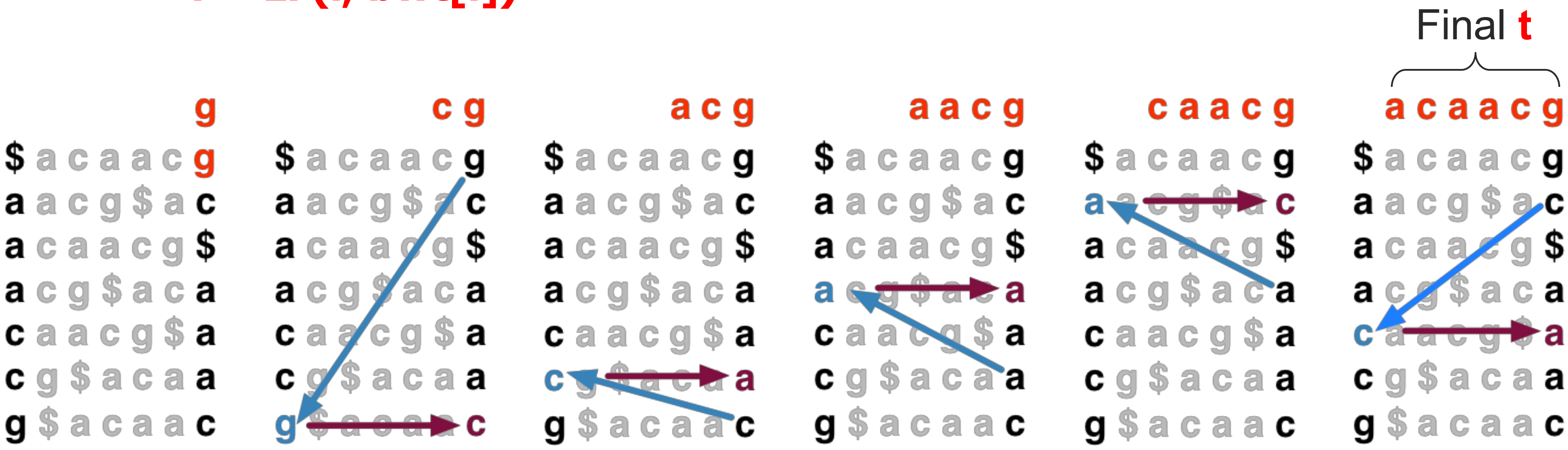
Count(idx, qc) – Number of qc characters before position idx in BWT(T)

Courtesy of Ben Langmead. Used with permission.

http://www.cbcb.umd.edu/~langmead/NCBI_Nov2008.ppt

The Walk Left Algorithm inverts the BWT

```
i = 0
t = ""
while bwt[i] != '$':
    t = bwt[i] + t
    i = LF(i, bwt[i])
```



Final t

```
            g             c g             a c g           a a c g         c a a c g       a c a a c g
$ a c a a c g   $ a c a a c g   $ a c a a c g   $ a c a a c g   $ a c a a c g   $ a c a a c g
a a c g $ a c   a a c g $ a c   a a c g $ a c   a a c g $ a c   a a c g $ a c   a a c g $ a c
a c a a c g $   a c a a c g $   a c a a c g $   a c a a c g $   a c a a c g $   a c a a c g $
a c g $ a c a   a c g $ a c a   a c g $ a c a   a c g $ a c a   a c g $ a c a   a c g $ a c a
c a a c g $ a   c a a c g $ a   c a a c g $ a   c a a c g $ a   c a a c g $ a   c a a c g $ a
c g $ a c a a   c g $ a c a a   c g $ a c a a   c g $ a c a a   c g $ a c a a   c g $ a c a a
g $ a c a a c   g $ a c a a c   g $ a c a a c   g $ a c a a c   g $ a c a a c   g $ a c a a c
```

Courtesy of Ben Langmead. Used with permission.

http://www.cbcb.umd.edu/~langmead/NCBI_Nov2008.ppt

Lecture 5 – Libraries and Indexing

- Library Complexity
  - How do we estimate the complexity of a sequencing library?

- Full-text Minute-size index (FM Index/BWT)
  - How do we convert a genome into an alternate representation that permits rapid matching of millions of sequence reads?

- Read Alignment
  - How can we use an FM index and BWT to rapidly align reads to a reference genome?

FM Index: querying

Look for range of rows of BWM(T) with *P* as prefix

Do this for *P*'s shortest suffix, then extend to successively longer suffixes until range becomes empty or we've exhausted *P*

P = **aba**

F						*L*
$\$$	a	b	a	a	b	a_0
a_0	$\$$	a	b	a	a	b_0
a_1	a	b	a	$\$$	a	b_1
a_2	b	a	$\$$	a	b	a_1
a_3	b	a	a	b	a	$\$$
b_0	a	$\$$	a	b	a	a_2
b_1	a	a	b	a	$\$$	a_3

FM Index: querying

Look for range of rows of BWM(T) with *P* as prefix

Do this for *P*'s shortest suffix, then extend to successively longer suffixes until range becomes empty or we've exhausted *P*

P = **aba**

Easy to find all the rows beginning with **a**, thanks to *F*'s simple structure

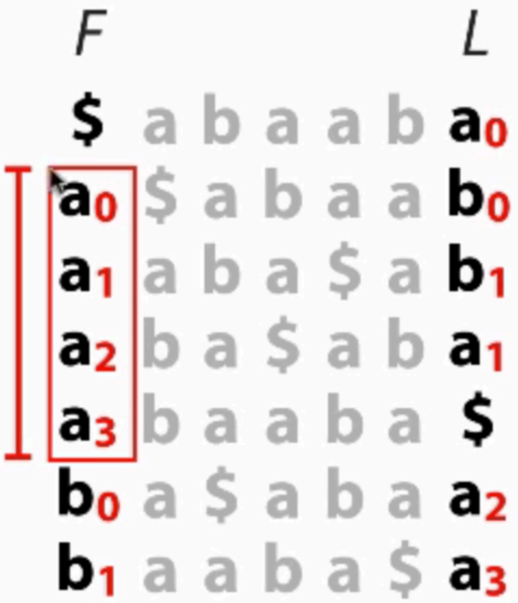
F						*L*
$\$$	a	b	a	a	b	**a_0**
a_0	$\$$	a	b	a	a	**b_0**
a_1	a	b	a	$\$$	a	**b_1**
a_2	b	a	$\$$	a	b	**a_1**
a_3	b	a	a	b	a	**$\$$**
b_0	a	$\$$	a	b	a	**a_2**
b_1	a	a	b	a	$\$$	**a_3**

FM Index: querying

Look for range of rows of BWM(T) with *P* as prefix

Do this for *P*'s shortest suffix, then extend to successively longer suffixes until range becomes empty or we've exhausted *P*

P = **aba**



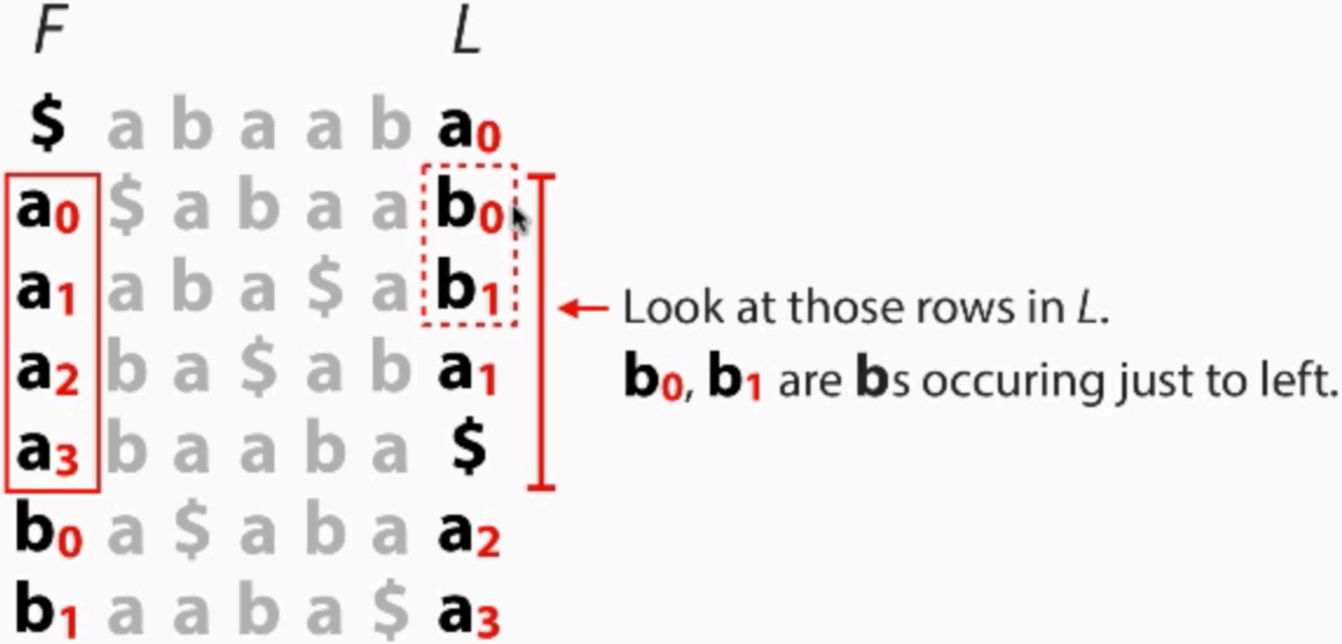
F						*L*
$	a	b	a	a	b	$\mathbf{a}_0$
$\mathbf{a}_0$	$	a	b	a	a	$\mathbf{b}_0$
$\mathbf{a}_1$	a	b	a	$	a	$\mathbf{b}_1$
$\mathbf{a}_2$	b	a	$	a	b	$\mathbf{a}_1$
$\mathbf{a}_3$	b	a	a	b	a	**$**
$\mathbf{b}_0$	a	$	a	b	a	$\mathbf{a}_2$
$\mathbf{b}_1$	a	a	b	a	$	$\mathbf{a}_3$

Easy to find all the rows beginning with **a**, thanks to *F*'s simple structure

FM Index: querying

We have rows beginning with **a**, now we seek rows beginning with **ba**

P = **aba**



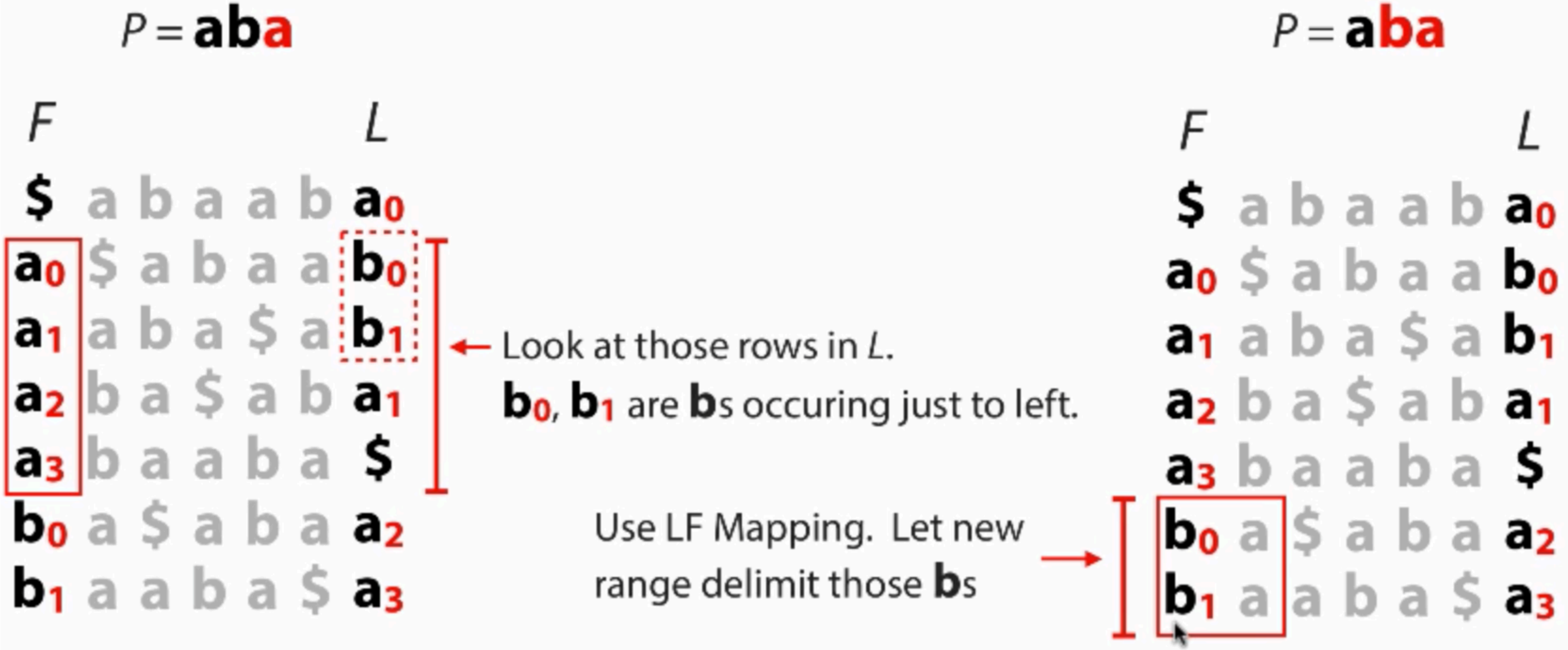
F						*L*
$	a	b	a	a	b	a_0
a_0	$	a	b	a	a	b_0
a_1	a	b	a	$	a	b_1
a_2	b	a	$	a	b	a_1
a_3	b	a	a	b	a	$
b_0	a	$	a	b	a	a_2
b_1	a	a	b	a	$	a_3

← Look at those rows in *L*.

b_0, b_1 are **b**s occuring just to left.

FM Index: querying

We have rows beginning with **a**, now we seek rows beginning with **ba**



$P =$ **aba**

F						L
\$	a	b	a	a	b	a_0
a_0	\$	a	b	a	a	b_0
a_1	a	b	a	\$	a	b_1
a_2	b	a	\$	a	b	a_1
a_3	b	a	a	b	a	\$
b_0	a	\$	a	b	a	a_2
b_1	a	a	b	a	\$	a_3

← Look at those rows in *L*.
b_0, b_1 are **b**s occuring just to left.

Use LF Mapping. Let new range delimit those **b**s →

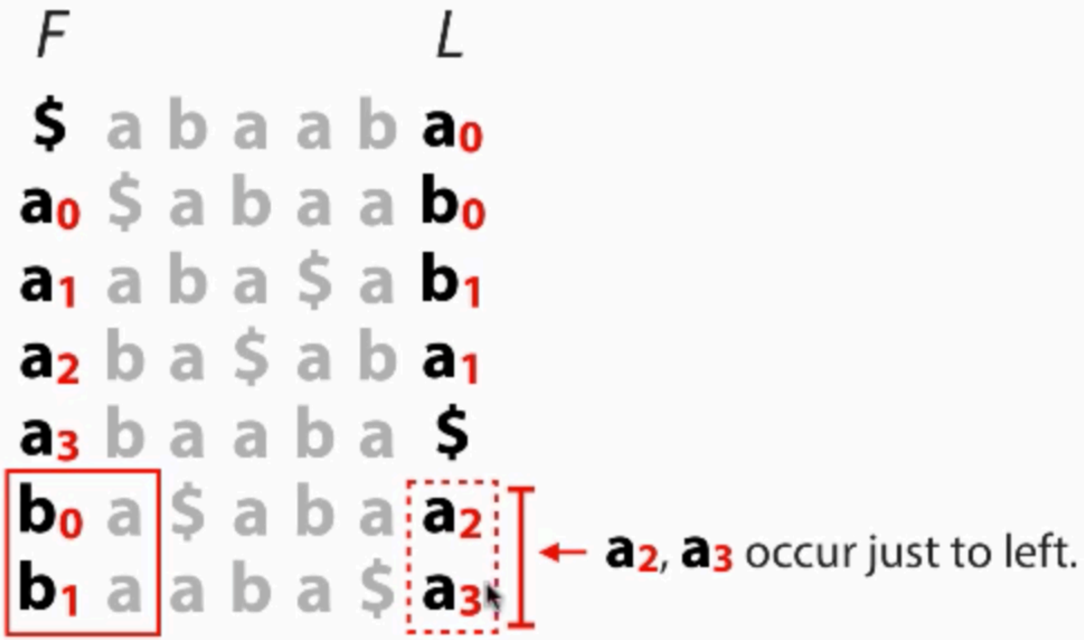
$P =$ **aba**

F						L
\$	a	b	a	a	b	a_0
a_0	\$	a	b	a	a	b_0
a_1	a	b	a	\$	a	b_1
a_2	b	a	\$	a	b	a_1
a_3	b	a	a	b	a	\$
b_0	a	\$	a	b	a	a_2
b_1	a	a	b	a	\$	a_3

FM Index: querying

We have rows beginning with **ba**, now we seek rows beginning with **aba**

$P =$ **aba**

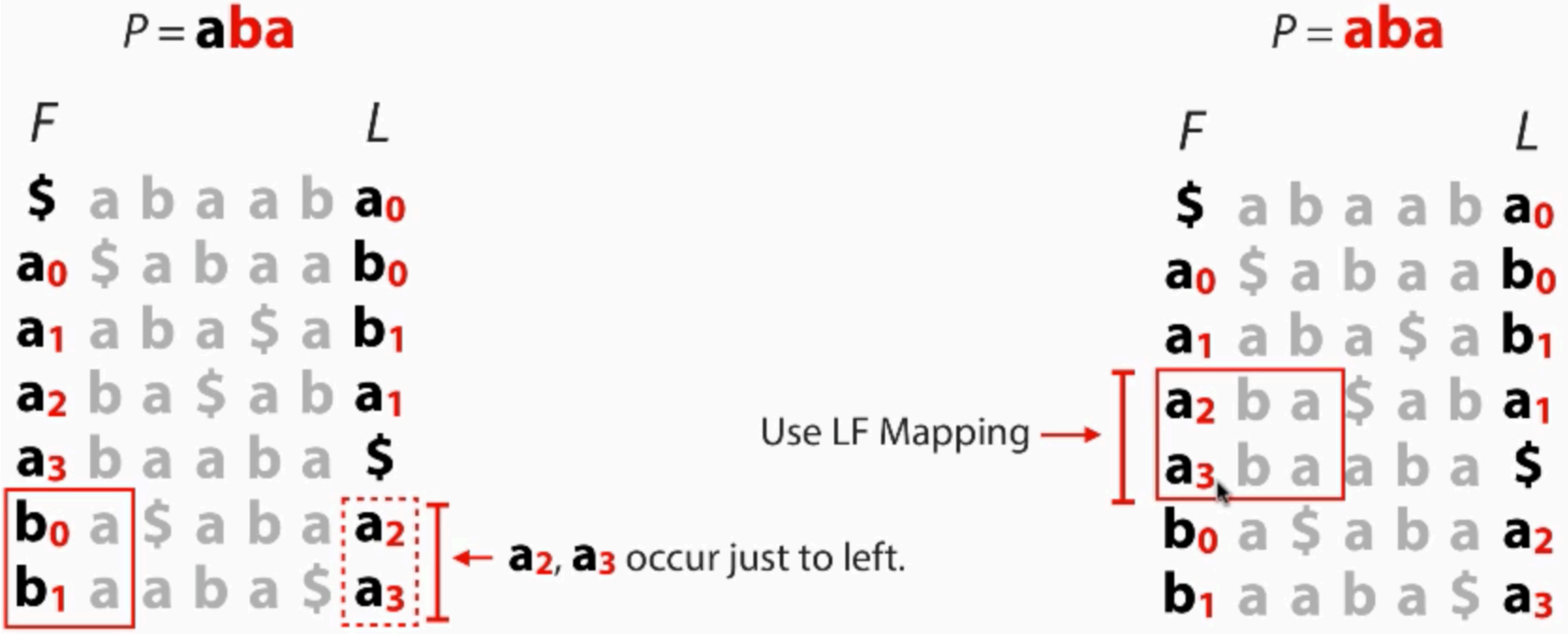


F						*L*
$	a	b	a	a	b	a_0
a_0	$	a	b	a	a	b_0
a_1	a	b	a	$	a	b_1
a_2	b	a	$	a	b	a_1
a_3	b	a	a	b	a	$
b_0	a	$	a	b	a	a_2
b_1	a	a	b	a	$	a_3

← a_2, a_3 occur just to left.

FM Index: querying

We have rows beginning with **ba**, now we seek rows beginning with **aba**



$P =$ **aba**

F						L
\$	a	b	a	a	b	a_0
a_0	\$	a	b	a	a	b_0
a_1	a	b	a	\$	a	b_1
a_2	b	a	\$	a	b	a_1
a_3	b	a	a	b	a	\$
b_0	a	\$	a	b	a	a_2
b_1	a	a	b	a	\$	a_3

← a_2, a_3 occur just to left.

$P =$ **aba**

F						L
\$	a	b	a	a	b	a_0
a_0	\$	a	b	a	a	b_0
a_1	a	b	a	\$	a	b_1
a_2	b	a	\$	a	b	a_1
a_3	b	a	a	b	a	\$
b_0	a	\$	a	b	a	a_2
b_1	a	a	b	a	\$	a_3

Use LF Mapping →