FM Index: Efficient matching with BWT

Ben Langmead

JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Department of Computer Science

Wavelet trees

Armed with Wavelet Trees, let's return to the Burrows-Wheeler Transform

We can reverse it efficiently now!

Burrows-Wheeler Transform

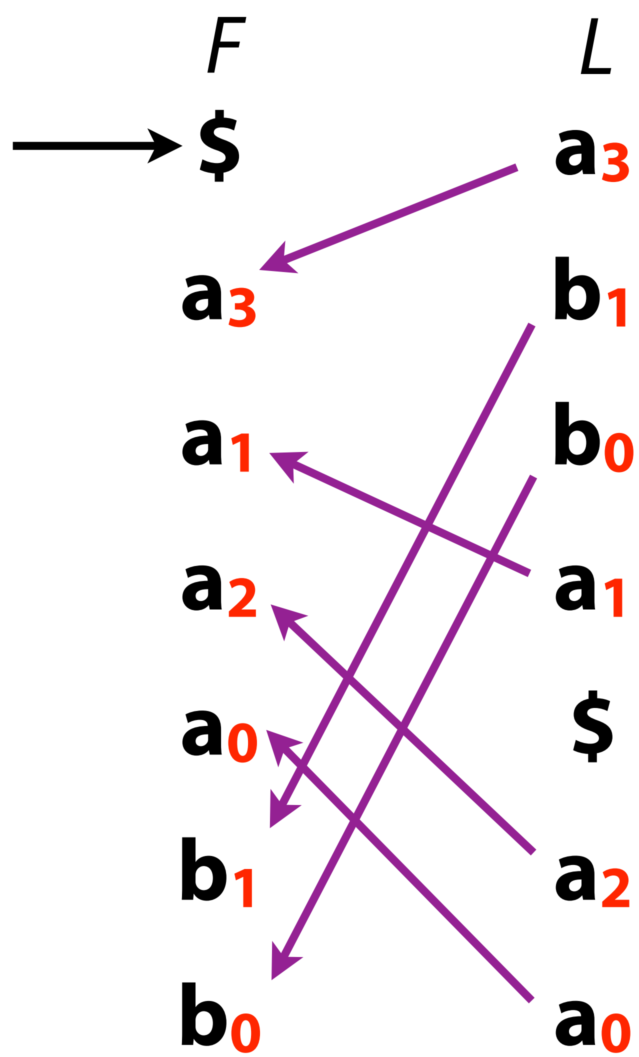
T: a_0 b_0 a_1 a_2 b_1 a_3 \$

	F	*L*
→	\$	a_3
	a_3	b_1
	a_1	b_0
	a_2	a_1
	a_0	\$
	b_1	a_2
	b_0	a_0

LF Mapping: The i^{th} occurrence of a character *c* in *L* and the i^{th} occurrence of *c* in *F* correspond to the *same* occurrence in *T*
(i.e. have same rank)

Burrows-Wheeler Transform

T: a_0 b_0 a_1 a_2 b_1 a_3 \$



LF Mapping: The i^{th} occurrence of a character *c* in *L* and the i^{th} occurrence of *c* in *F* correspond to the *same* occurrence in *T*
(i.e. have same rank)

Burrows-Wheeler Transform

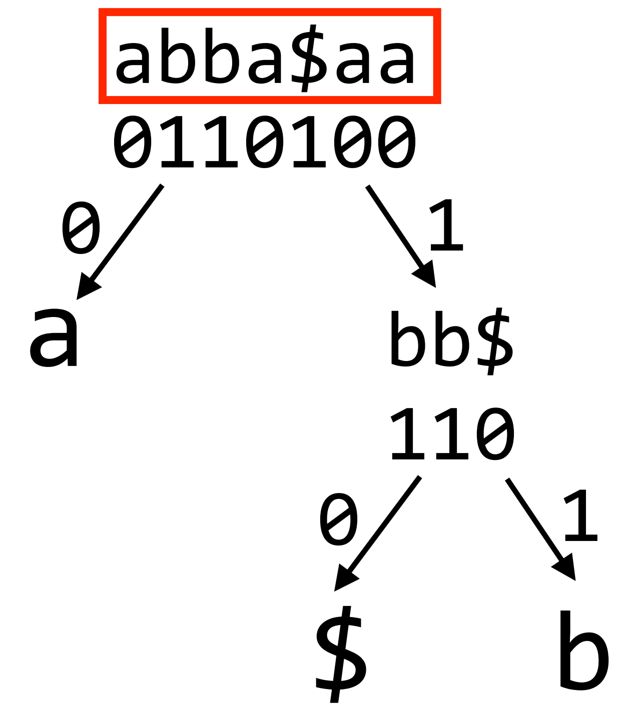
abba$aa

F	*L*
$	**a**
a	**b**
a	**b**
a	**a**
a	**$**
b	**a**
b	**a**

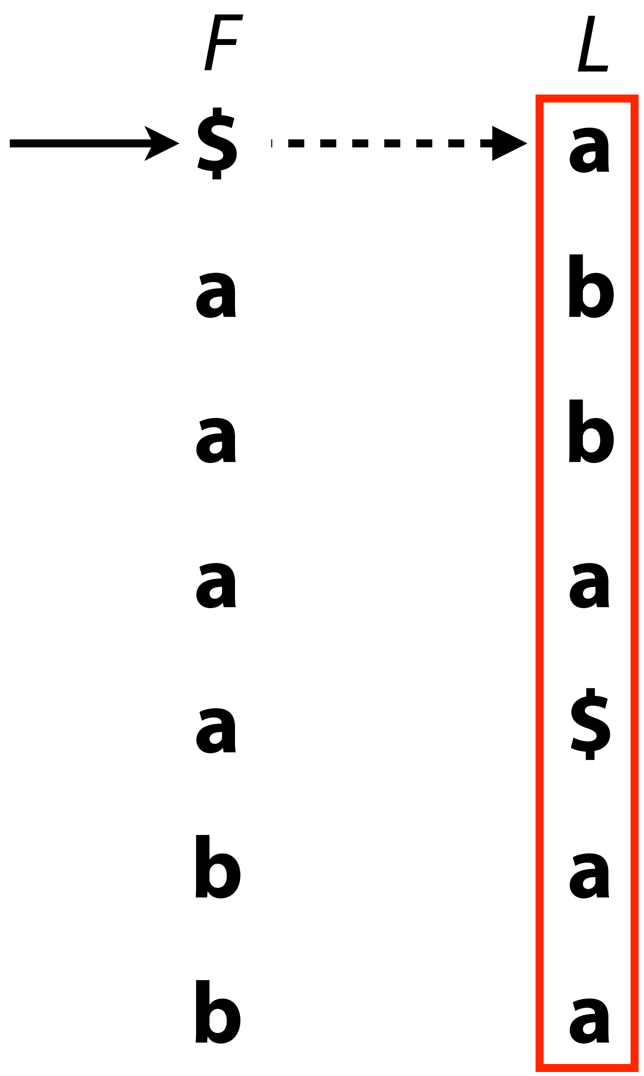
Burrows-Wheeler Transform

F	*L*
$	**a**
a	**b**
a	**b**
a	**a**
a	**$**
b	**a**
b	**a**

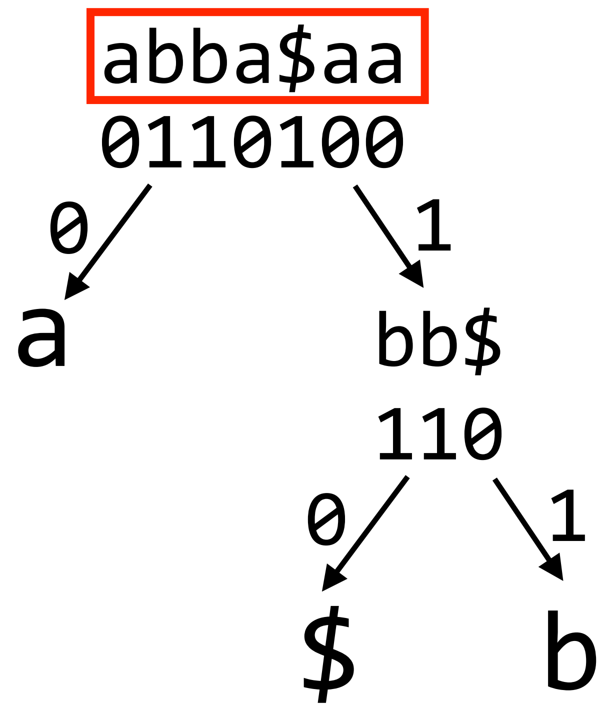
$\mathrm{WT}(L) =$



Burrows-Wheeler Transform



WT(L) =



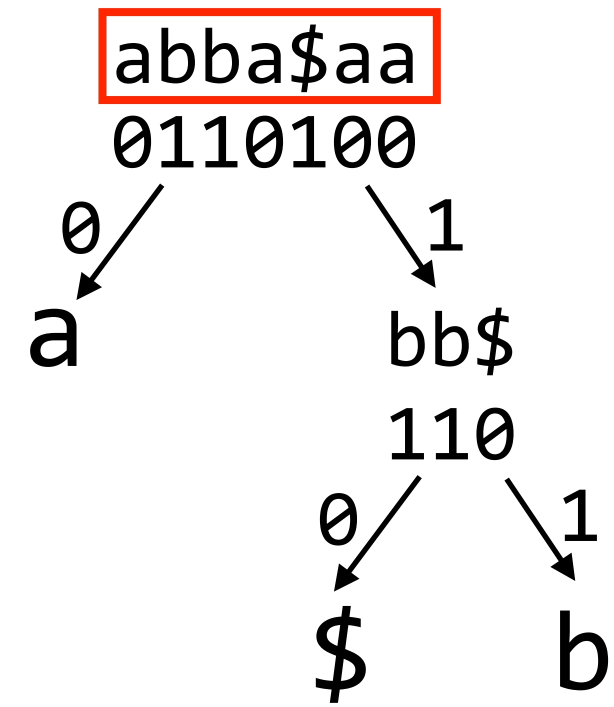
Recall: 1st row has $ in F, so start there

In L, we see an **a**. What is its rank?

Burrows-Wheeler Transform

F	L
$	**a**
a	**b**
a	**b**
a	**a**
a	**$**
b	**a**
b	**a**

$\mathrm{WT}(L) =$



Recall: 1st row has $ in F, so start there

In L, we see an **a**. What is its rank?

$\mathrm{WT}(L)\,.\,\mathrm{rank}_{\mathrm{a}}(0) \;= 0$

Burrows-Wheeler Transform

F	*L*	Rank	Skip	Next row	Next char
$	**a**				**a**
a	**b**				
a	**b**				
a	**a**				
a	**$**				
b	**a**				
b	**a**				

Burrows-Wheeler Transform

F	*L*
\$	**a**
a	**b**
a	**b**
a	**a**
a	**\$**
b	**a**
b	**a**

Rank	Skip	Next row	Next char
			a
$L.\mathrm{rank}_{\mathrm{a}}(0) = 0$	$1 \times \$ = 1$	$0 + 1 = 1$	**b**

Burrows-Wheeler Transform

F	*L*
$	**a**
a	**b**
a	**b**
a	**a**
a	**$**
b	**a**
b	**a**

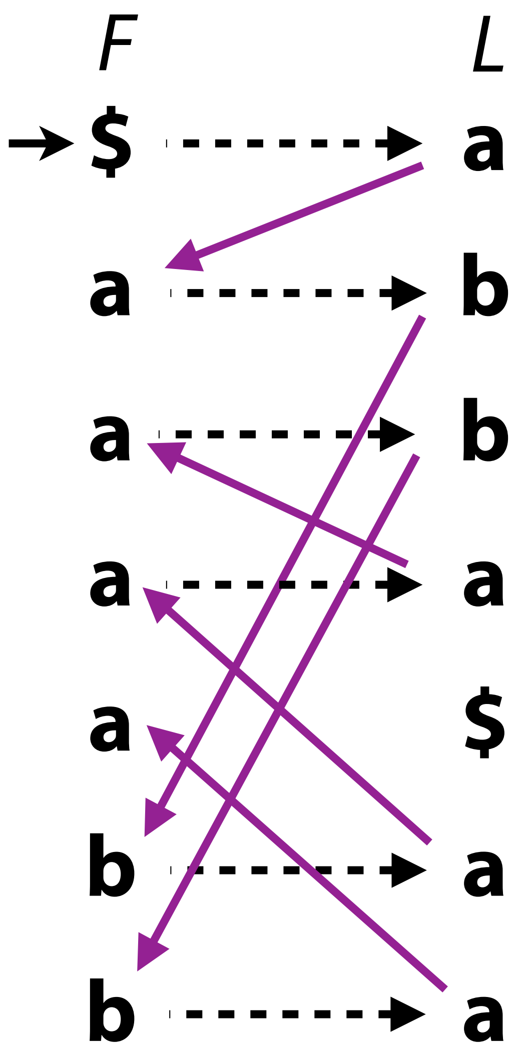
Rank	Skip	Next row	Next char
			a
$L.\mathrm{rank}_{\mathrm{a}}(0) = 0$	1 x $ = 1	0 + 1 = 1	**b**
$L.\mathrm{rank}_{\mathrm{b}}(1) = 0$	1 x $ + 4 x **a** = 5	0 + 5 = 5	**a**

Burrows-Wheeler Transform

F	*L*
$	**a**
a	**b**
a	**b**
a	**a**
a	**$**
b	**a**
b	**a**

Rank	Skip	Next row	Next char
			a
$L.\text{rank}_{\text{a}}(0) = 0$	1 x \$ = 1	0 + 1 = 1	**b**
$L.\text{rank}_{\text{b}}(1) = 0$	1 x \$ + 4 x **a** = 5	0 + 5 = 5	**a**
$L.\text{rank}_{\text{a}}(5) = 2$	1 x \$ = 1	2 + 1 = 3	**a**

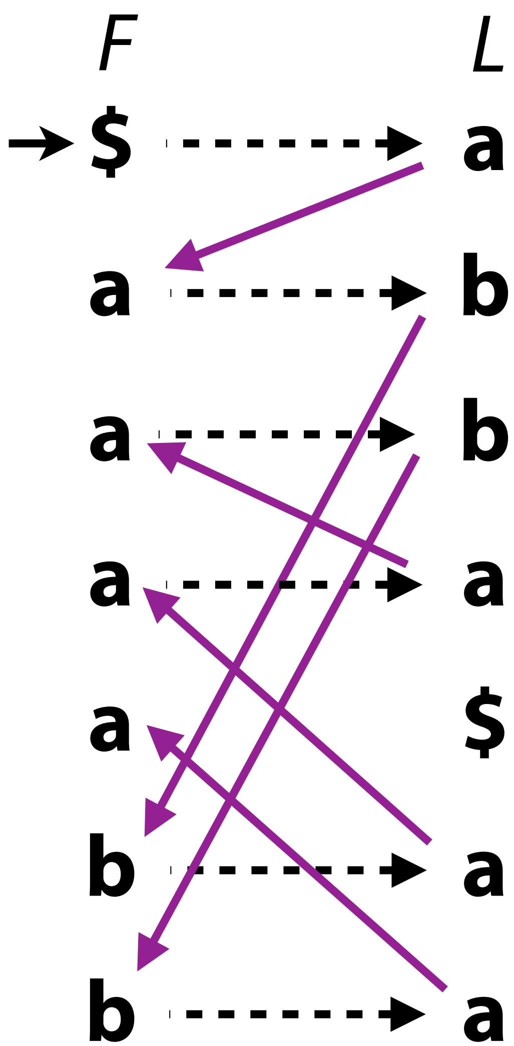
Burrows-Wheeler Transform



Rank	Skip	Next row	Next char
			a
$L.\text{rank}_a(0) = 0$	$1 \times \$ = 1$	$0 + 1 = 1$	**b**
$L.\text{rank}_b(1) = 0$	$1 \times \$ + 4 \times \mathbf{a} = 5$	$0 + 5 = 5$	**a**
$L.\text{rank}_a(5) = 2$	$1 \times \$ = 1$	$2 + 1 = 3$	**a**
$L.\text{rank}_a(3) = 1$	$1 \times \$ = 1$	$1 + 1 = 2$	**b**
$L.\text{rank}_b(2) = 1$	$1 \times \$ + 4 \times \mathbf{a} = 5$	$1 + 5 = 6$	**a**
$L.\text{rank}_a(6) = 3$	$1 \times \$ = 1$	$3 + 1 = 4$	**$**

Skip amount can be looked up; pre-calculate C where $C[c]$ (c is a character) equals the number of characters alphabetically smaller than c in T

Burrows-Wheeler Transform



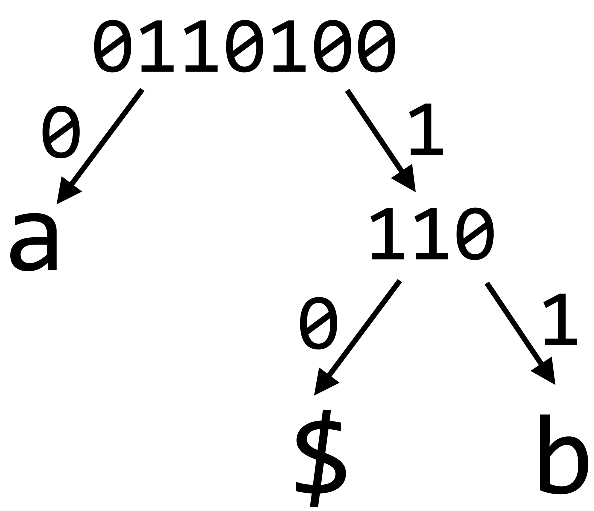
Rank	Skip	Next row	Next char
			a
$L.\text{rank}_a(0) = 0$	1 x \$ = 1	0 + 1 = 1	b
$L.\text{rank}_b(1) = 0$	1 x \$ + 4 x **a** = 5	0 + 5 = 5	a
$L.\text{rank}_a(5) = 2$	1 x \$ = 1	2 + 1 = 3	a
$L.\text{rank}_a(3) = 1$	1 x \$ = 1	1 + 1 = 2	b
$L.\text{rank}_b(2) = 1$	1 x \$ + 4 x **a** = 5	1 + 5 = 6	a
$L.\text{rank}_a(6) = 3$	1 x \$ = 1	3 + 1 = 4	\$

Skip amount can be looked up; pre-calculate C where $C[c]$ (c is a character) equals the number of characters alphabetically smaller than c in T

Here, $C[\$] = 0,\ C[\mathtt{a}] = 1,\ C[\mathtt{b}] = 5$

Burrows-Wheeler Transform

$\mathrm{WT}(\mathrm{BWT}(T))$



Rank	Skip	Next row	Next char
			a
$L.\mathrm{rank}_a(0) = 0$	1 x \$ = 1	0 + 1 = 1	**b**
$L.\mathrm{rank}_b(1) = 0$	1 x \$ + 4 x **a** = 5	0 + 5 = 5	**a**
$L.\mathrm{rank}_a(5) = 2$	1 x \$ = 1	2 + 1 = 3	**a**
$L.\mathrm{rank}_a(3) = 1$	1 x \$ = 1	1 + 1 = 2	**b**
$L.\mathrm{rank}_b(2) = 1$	1 x \$ + 4 x **a** = 5	1 + 5 = 6	**a**
$L.\mathrm{rank}_a(6) = 3$	1 x \$ = 1	3 + 1 = 4	**\$**

Reversing is $O\left(\underbrace{n}_{\text{steps}} \underbrace{\log_2 \sigma}_{\text{rank query}} \right)$

Rank + skip = LF mapping

Burrows-Wheeler Transform

Principles of navigation

Use WT(BWT(T)) to reverse: BWT(T) $\rightarrow T$

How do we do *indexing*?

Indexing

When we add some auxiliary data structures to make it easier to answer indexing queries

Opportunistic Data Structures with Applications

Paolo Ferragina*
Università di Pisa

Giovanni Manzini†
Università del Piemonte Orientale

"FM Index"

Abstract

In this paper we address the issue of compressing and indexing data. We devise a data structure whose space occupancy is a function of the entropy of the underlying data set. We call the data structure opportunistic *since its space occupancy is decreased when the input is compressible and this space reduction is achieved at no significant slowdown in the query performance. More precisely, its space occupancy is optimal in an information-content sense because a text $T[1,u]$ is stored using $O(H_k(T)) + o(1)$ bits per input symbol in the worst case, where $H_k(T)$ is the kth order empirical entropy of T (the bound holds for any fixed k). Given an arbitrary string $P[1,p]$, the opportunistic data structure allows to search for the occ occurrences of P in T in $O(p + occ \log^{\epsilon} u)$ time (for any fixed $\epsilon > 0$). If data are*

Ferragina, Paolo, and Giovanni Manzini. "Opportunistic data structures with applications." *Proceedings 41st Annual Symposium on Foundations of Computer Science*. IEEE, 2000.

Indexing

A ***full-text index*** for text $T \in \Sigma^n$ is a structure giving efficient answers to queries:

Locate(P), where $P \in \Sigma^m$, returns all offsets where P matches a substring of T

Count(P) returns # of offsets where P matches a substring of T

Extract(i, m) returns $T[i : i + m - 1]$ (length-m substring starting at i)

Indexing

A ***full-text index*** for text $T \in \Sigma^n$ is a structure giving efficient answers to queries:

Locate(P), where $P \in \Sigma^m$, returns all offsets where P matches a substring of T

Count(P) returns # of offsets where P matches a substring of T

Extract(i, m) returns $T[i : i + m - 1]$ (length-m substring starting at i)

FM Index: querying

How to ***find***, ***count*** and ***locate*** substrings matching a query?

$	a	b	a	a	b	**a**
a	$	a	b	a	a	**b**
a	a	b	a	$	a	**b**
a	b	a	$	a	b	**a**
a	b	a	a	b	a	**$**
b	a	$	a	b	a	**a**
b	a	a	b	a	$	**a**

FM Index: querying

Observation 1: Rows with same prefix are consecutive

$	a	b	a	a	b	**a**
a	$	a	b	a	a	**b**
a	a	b	a	$	a	**b**
a	b	a	$	a	b	**a**
a	b	a	a	b	a	**$**
b	a	$	a	b	a	**a**
b	a	a	b	a	$	**a**

Observation 2: Characters in last column are those *preceding* the prefixes (to their *left* in T)

FM Index: querying

Given pattern P, $|P| = m$, start with shortest suffix of and match successively longer suffixes

$P =$ **aba**

Easy to find all the rows beginning with **a**

$[C[\mathrm{a}], C[\mathrm{b}]) = [1{,}5)$

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

Subscripts are ranks in L

FM Index: querying

We have rows beginning with **a**, now we want 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

Now we have the rows with prefix **ba**

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.

Use LF Mapping →

$P =$ **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

Now we have the rows with prefix **aba**

$$T.\text{count}(\text{aba}) = 2$$

FM Index: querying

When *P* does not occur in *T*, we eventually fail to find next character in *L*:

P = **bba**

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	No **b**s!
b_1	a	a	b	a	$\$$	a_3	

Rows with **ba** prefix

FM Index: querying

$P = $ **ab**a

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**

Next char	Rank	Skip	Next range
a		1 x \$ = 1 1 x \$ + 5 x **a** = 5	1 5

FM Index: querying

$P = \mathbf{aba}$

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

Next char	Rank	Skip	Next range
a		1 x $ = 1 1 x $ + 5 x **a** = 5	1 5
b	$L.\text{rank}_b(1) = 0$ $L.\text{rank}_b(5) = 2$	1 x $ + 5 x **a** = 5	0 + 5 = 5 2 + 5 = 7

FM Index: querying

$P =$ **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

Next char	Rank	Skip	Next range
a		1 x \$ = 1 1 x \$ + 5 x **a** = 5	1 5
b	$L.\text{rank}_b(1) = 0$ $L.\text{rank}_b(5) = 2$	1 x \$ + 5 x **a** = 5	0 + 5 = 5 2 + 5 = 7
a	$L.\text{rank}_a(5) = 2$ $L.\text{rank}_a(7) = 4$	1 x \$ = 1	0 + 1 = 3 2 + 1 = 5

FM Index: querying

$P =$ **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	\$	$\mathbf{a}_3$

$T.\text{count}(\text{aba}) = 2$

Next char	Rank	Skip	Next range
a		1 x \$ = 1 1 x \$ + 5 x **a** = 5	1 5
b	$L.\text{rank}_\text{b}(1) = 0$ $L.\text{rank}_\text{b}(5) = 2$	1 x \$ + 5 x **a** = 5	0 + 5 = 5 2 + 5 = 7
a	$L.\text{rank}_\text{a}(5) = 2$ $L.\text{rank}_\text{a}(7) = 4$	1 x \$ = 1	0 + 1 = 3 2 + 1 = 5

FM Index: querying

FM index match(P):

Given query string P

top $\leftarrow 0$

bot $\leftarrow |T|$

$i \leftarrow |P| - 1$

while $i \geq 0$ and bot $>$ top

  $c \leftarrow P[i]$

  top $\leftarrow$ BWT . $C[c]$ + BWT . $\text{rank}_c(\text{top})$

  bot $\leftarrow$ BWT . $C[c]$ + BWT . $\text{rank}_c(\text{bot})$

  $i \leftarrow i - 1$

return (top, bot)

Skip Rank

Rank

(For simplicity, version starts with the all-inclusive range rather than using 2 initial BWT . C[. . .] lookups to get the range for the length-1 suffix)

FM Index: querying

A ***full-text index*** for text $T \in \Sigma^n$ is a structure giving efficient answers to queries:

$\boxed{\text{Locate}(P)\text{, where } P \in \Sigma^m\text{, returns all offsets where } P \text{ matches a substring of } T}$

Count(P) returns # of offsets where P matches a substring of T

Extract(i, m) returns $T[i : i + m - 1]$ (length-m substring starting at i)

FM Index: querying

F						*L*
$	a	b	a	a	b	**a**
a	$	a	b	a	a	**b**
a	a	b	a	$	a	**b**
a	b	a	$	a	b	**a**
a	b	a	a	b	a	**$**
b	a	$	a	b	a	**a**
b	a	a	b	a	$	**a**

Where are these occurrences in *T*?

FM Index: querying

Where are these occurrences in *T*?

$	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

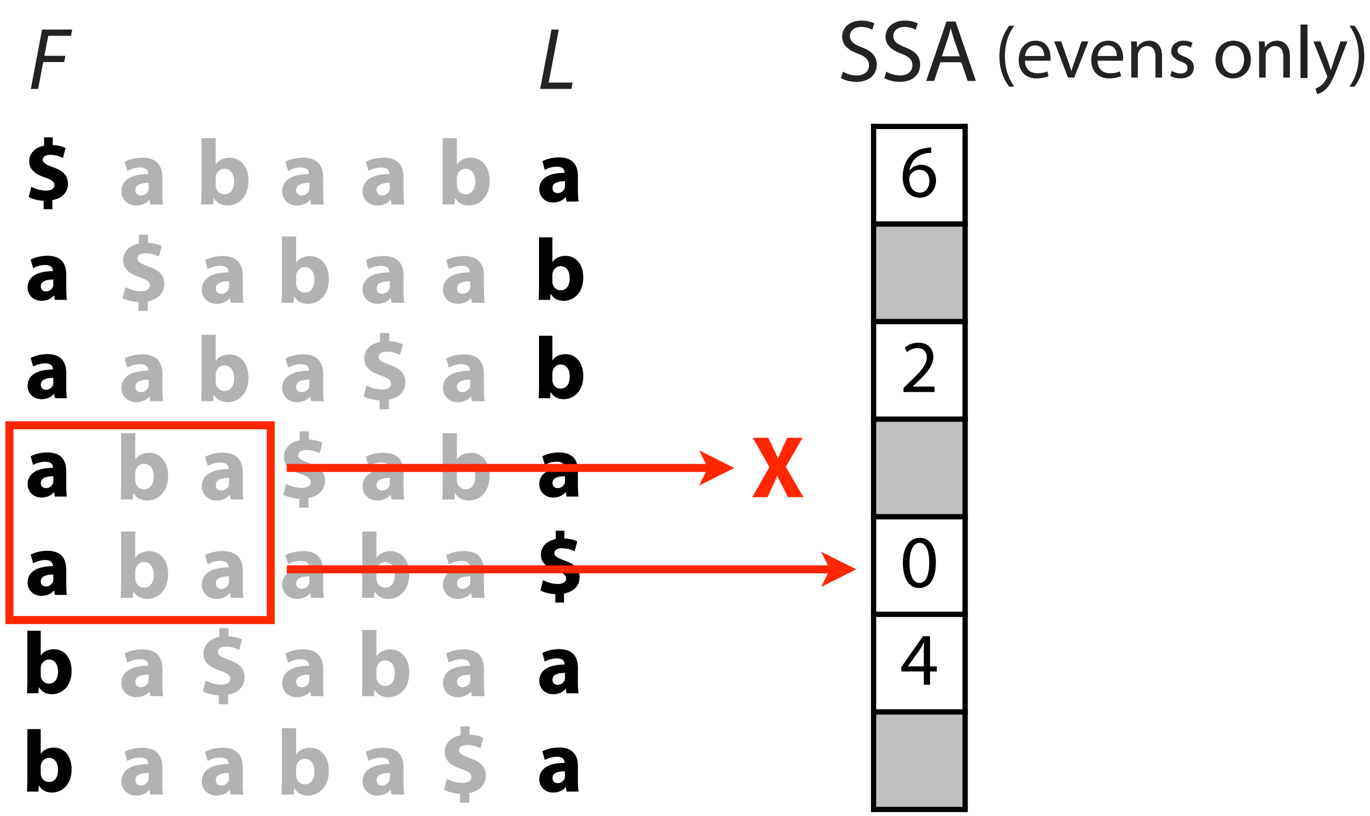
If we had suffix array, we could look up offsets...

F						*L*	SA	
$	a	b	a	a	b	a	6	$
a	$	a	b	a	a	b	5	a $
a	a	b	a	$	a	b	2	a a b a $
a	b	a	$	a	b	a	3	a b a $
a	b	a	a	b	a	$	0	a b a a b a $
b	a	$	a	b	a	a	4	b a $
b	a	a	b	a	$	a	1	b a a b a $

Offsets: 0, 3

FM Index: resolving offsets

Sampled Suffix Array (SSA): store some suffix array elements, not all



Lookup for row 4 succeeds

Lookup for row 3 fails - SA entry was discarded