String Algorithms and Data Structures

FM Index

CS 199-225
Brad Solomon

October 31, 2022
Burrows-Wheeler Transform

Reversible permutation of the characters of a string

Burrows-Wheeler Transform: LF Mapping

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

Any ranking we give to characters in $T$ will match in $F$ and $L$. 
Burrows-Wheeler Transform: LF Mapping

Another way to visualize:

$T: \ a_0 \ b_0 \ a_1 \ a_2 \ b_1 \ a_3 \ \$
A review of ‘F’ and ‘L’

\[ L = \text{CGGGCC}$ \quad \Sigma = \text{“ACGT”} \]

How can we represent \( F \)?

As a full text string: \( F = \$\text{CCCCGGG} \)

As a map<\text{string, int}>: \( F = \{\$': 1, 'C': 3, 'G': 3\} \)

As a vector<int>: \( F = [0, 3, 3, 0] \)
A review of ‘F’ and ‘L’

BWT(T) = e$1ppa

What row index in $F$ contains ‘e’?

What row index in $L$ contains ‘e’?

What row index in $F$ contains the second ‘p’?
FM Index

An index combining the BWT with a few small auxiliary data structures

Core of index is first (F) and last (L) rows from BWM:

\[ \begin{array}{cccc}
F & L \\
$ & a & b & a & b & a \\
a & $ & a & b & a & a & b \\
a & a & b & a & $ & a & b \\
a & b & a & $ & a & b & a \\
a & a & b & a & b & a & $ \\
b & a & $ & a & b & a & a \\
b & a & a & b & a & $ & a \\
\end{array} \]

L is the same size as T

\( F \) can be represented as array of \( |\Sigma| \) integers (or not stored at all!)

We’re discarding \( T \) — we can recover it from \( L \)!
FM Index: Querying

\[ P = A \ A \ A \]

\[
\begin{array}{cccccccc}
$ & B & B & B & A & A & A_0 \\
A_0 & $ & B & B & B & A & A_1 \\
A_1 & A & $ & B & B & B & A_2 \\
A_2 & A & A & $ & B & B & B_0 \\
B_0 & A & A & A & $ & B & B_1 \\
B_1 & B & A & A & A & $ & B_2 \\
\end{array}
\]
FM Index: Querying

\[ P = \text{B A B} \]

\[
\begin{array}{ccccccc}
$ & B & B & B & A & A & A_0 \\
A_0 & $ & B & B & B & A & A_1 \\
A_1 & A & $ & B & B & B & A_2 \\
A_2 & A & A & $ & B & B & B_0 \\
B_0 & A & A & A & $ & B & B_1 \\
B_1 & B & A & A & A & $ & B_2 \\
\end{array}
\]
FM Index: Lingering Issues

(1) Scanning for preceding character in \( L \) is slow

\[
\begin{array}{cccccccc}
\$ & a & b & a & a & b & a_0 \\
a_0 & \$ & a & b & a & a & b_0 \\
a_1 & a & b & a & $ & a & b_1 \\
a_2 & b & a & $ & a & b & a_1 \\
a_3 & b & a & b & a & $ \\
b_0 & a & $ & a & b & a & a_2 \\
b_1 & a & a & b & a & $ & a_3 \\
\end{array}
\]

We don’t store ranks!

(2) Need way to find where matches occur in \( T \):

\[
\begin{array}{cccccccc}
\$ & a & b & a & a & b & a_0 \\
a_0 & \$ & a & b & a & a & b_0 \\
a_1 & a & b & a & $ & a & b_1 \\
a_2 & b & a & $ & a & b & a_1 \\
a_3 & b & a & a & b & a & $ \\
b_0 & a & $ & a & b & a & a_2 \\
b_1 & a & a & b & a & $ & a_3 \\
\end{array}
\]

Current output: \([3,4]\)

Location in \( T \): \([0,3]\)

This is where our auxiliary data structures come in…
FM Index: Fast rank calculations

Is there a fast way to determine which specific $bs$ precede the $as$ in our range?

```
$ 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
```

More generally, given a range in $L$ and a character to search, how can we quickly find all matches (and their ranks)?
FM Index: Occurrence Table

Idea: pre-calculate cumulative # `a` s, `b` s in $L$ up to every row:

<table>
<thead>
<tr>
<th>$L$</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$$</td>
<td>$</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FM Index: Occurrence Table

Idea: pre-calculate cumulative # a's, b's in \( L \) up to every row:

<table>
<thead>
<tr>
<th>( L )</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>a</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>a</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>
FM Index: Occurrence Table

Idea: pre-calculate cumulative # a's, b's in L up to every row:

| $ | a | a | 1 | 0 |
| a | b | b | 1 | 1 |
| a | b | b | 1 | 2 |
| a | a | a | 2 | 2 |
| a | $ | a | 2 | 2 |
| b | a | a | 3 | 2 |
| b | a | a | 4 | 2 |

Query: 'aba'
FM Index: Occurrence Table

Idea: pre-calculate cumulative # a's, b's in L up to every row:

<table>
<thead>
<tr>
<th>F</th>
<th>L</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>a</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>a</td>
<td>$</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>b</td>
<td>a</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>b</td>
<td>a</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Query: ‘aba’

- 0 b's up to & including this row
- 2 b's up to & including this row
FM Index: Occurrence Table

Idea: pre-calculate cumulative # a's, b's in $L$ up to every row:

<table>
<thead>
<tr>
<th>$F$</th>
<th>$L$</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$</td>
<td>a</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>a</td>
<td>$$</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>b</td>
<td>a</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>b</td>
<td>a</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Query: ‘aba’
What two indices should I look up? What ranks did we find?

<table>
<thead>
<tr>
<th>$</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>b</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>
FM Index: Occurrence Table

An index combining the BWT with a few small auxiliary data structures

Occurrence table speeds up $L$ lookup by implicitly storing ranks

Scan is $O(m)$ work

Lookup is $O(1)$ work

Table is $m \times |\Sigma|$ integers — that’s worse than a suffix array!
Next idea: pre-calculate # of `a`s, `b`s in `L` up to some rows, e.g. every 5th row. Call pre-calculated rows *checkpoints*.

<table>
<thead>
<tr>
<th>$F$</th>
<th>$L$</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$,</td>
<td><code>a</code></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td><code>a</code></td>
<td><code>b</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>a</code></td>
<td><code>b</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>a</code></td>
<td><code>b</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>a</code></td>
<td><code>a</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>a</code></td>
<td><code>$\$</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>b</code></td>
<td><code>a</code></td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td><code>b</code></td>
<td><code>a</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FM Index: Occurrence Table

To resolve a lookup for a non-checkpoint row, walk to nearest checkpoint. Use value at that checkpoint, *adjusted for characters we saw along the way.*

<table>
<thead>
<tr>
<th>F</th>
<th>L</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>a</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>a</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>b</td>
<td>a</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
If checkpoints are $O(1)$ distance apart, lookups are $O(1)$.

What goes here?
$482 + 2 = 484$

What’s goes here?
$439 - 2 = 437$

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>482</td>
<td>432</td>
<td></td>
</tr>
<tr>
<td>488</td>
<td>439</td>
<td></td>
</tr>
</tbody>
</table>
FM Index: Occurrence Table

An index combining the BWT with a few small auxiliary data structures

Occurrence table speeds up $L$ lookup by implicitly storing ranks

Checkpoints reduce the storage costs (Still $O(m)$ but better than SA)
FM Index: Querying

Problem 2: We don’t know *where* the matches are in \( T \)...
FM Index: Suffix Array Sampling

Idea: store some suffix array elements, but not all

<table>
<thead>
<tr>
<th>F</th>
<th>L</th>
<th>SA' (evens only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ a b a a b a</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>a $ a b a a b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a a b a $ a b</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>a b a $ a b a</td>
<td>X X X X X</td>
<td>0</td>
</tr>
<tr>
<td>a b a a b a $</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b a $ a b a a</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>b a a b a $ a</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Lookup for row 4 succeeds

Lookup for row 3 fails - SA entry was discarded
LF Mapping tells us that “a” at the end of row 3 corresponds to “a” at the beginning of row 2.

If saved SA values are $O(1)$ positions apart in $T$, resolving index is $O(1)$ time.
FM Index: Suffix Array Sampling

Many LF-mapping steps may be required to get to a sampled row:

\[ F \quad L \quad SA' \text{ (every 4th)} \]

Starting here →

\[
\begin{array}{ccccccc}
\$ & a & b & a & a & b & a \\
a & \$ & a & b & a & a & b \\
a & a & b & a & \$ & a & b \\
a & b & a & \$ & a & b & a \\
b & a & a & b & a & \$ \\
b & a & a & b & a & \$ & a \\
\end{array}
\]
FM Index: Suffix Array Sampling

Many LF-mapping steps may be required to get to a sampled row:

\[
\begin{array}{ccc}
F & L & \text{SA'} (\text{every 4th}) \\
$ & a & b & a & a & b & a \\
a & $ & a & b & a & a & b \\
a & a & b & a & $ & b & b & a \\
a & b & a & b & a & a & b & a \\
b & a & a & b & a & $ & a \\
\end{array}
\]
FM Index: Suffix Array Sampling

Many LF-mapping steps may be required to get to a sampled row:
FM Index: Suffix Array Sampling

Many LF-mapping steps may be required to get to a sampled row:

$$\begin{array}{c|c|c}
F & L & SA' (\text{every 4th}) \\
\$ & a & a \\
a & \$ & a \\
a & a & \$ \\
a & a & b \\
a & \$ & b \\
0 & 0 & 4 \\
4 & 4 & 4 \\
3 & 3 & 3 \\
\end{array}$$

Missing value = 0 (SA val at destination) + 3 (# steps to destination) = 3
FM Index: Suffix Array Sampling

An index combining the BWT with *a few small auxiliary data structures*

Stores all index positions in T with O(1) extra work to calculate

Lets put all these pieces together…
FM Index: Querying

\[ P = \text{aba} \]

\[
\begin{array}{c|cc|cc|cc}
F & a & b & a & a & b & a_0 \\
\hline
a_0 & $ & a & b & a & a & b \\
a_1 & a & b & a & $ & a & b \\
a_2 & b & a & $ & a & b & a_1 \\
a_3 & b & a & a & b & a & $ \\
\end{array}
\]

```
get_frange()
```
pair<int, int> get_frange(string c, int s, int e)

Input:

- **string c**: The char we are looking for in $F$
- **int s**: The starting rank value
- **int e**: The ending rank value

Output:

- A pair of values (index start, index end)

What are $c$, $s$, and $e$?

What are the output values?
FM Index: Querying

$p = aba$

`get_coverage()`

`get_revert()`

$F$

$L$

$\$

$a$

$b$

$a$

$a$

$a$

$b$

$a$

$0$

$1$

$2$

$3$

`get_frange()`

`get_lrange()`
pair<int, int> get_lrange(string c, int s, int e)

Input:
- **string c**: The char we are looking for in \( F \)
- **int s**: The starting **index** of our range
- **int e**: The ending **index** of our range

Output:
- A pair of values (# occurrences start, end)

What are c, s, and e?

What are the output values?
FM Index: Querying

$p = aba$

$F \quad L$

\begin{align*}
$ & a \ b \ a \ a \ b \ a_0 \\
\text{a}_0 & \$ \ a \ b \ a \ a \ b \\
\text{a}_1 & a \ b \ a \$ \ a \ b \\
\text{a}_2 & b \ a \$ \ a \ b \ a_1 \\
\text{a}_3 & b \ a \ a \ b \ a \$ \\
\text{b} & a \$ \ a \ b \ a \text{a}_2 \\
\text{b} & a \ a \ b \ a \$ \text{a}_3
\end{align*}$

get_frange() \quad \text{get_lrange()}
pair<int, int> get_frange(string c, int s, int e)

Input:
- **string c**: The char we are looking for in $F$
- **int s**: The starting **rank** value
- **int e**: The ending **rank** value

Output:
- A pair of values (index start, index end)

What are c, s, and e?

What are the output values?
FM Index: Querying

$P = \texttt{aba}$

$F \quad L$

\begin{array}{llllll}
\$ & a & b & a & a & b & a_0 \\
 a_0 & $ & a & b & a & a & b \\
 a_1 & a & b & a & $ & a & b \\
 a_2 & b & a & $ & a & b & a_1 \\
 a_3 & b & a & a & b & a & $ \\
 b & a & $ & a & b & a_2 \\
 b & a & a & b & a & $ & a_3 \\
\end{array}$

get_frange() \quad get_lrange()
$P = \text{aba}$

\[
\begin{array}{ll}
F & L \\
\$ & a \ b \ a \ a \ b \ a_0 \\
a_0 & $ \ a \ b \ a \ a \ b_0 \\
a_1 & a \ b \ a \ $ \ a \ b_1 \\
a_2 & b \ a \ $ \ a \ b \ a_1 \\
a_3 & b \ a \ a \ b \ a \ $ \\
b_0 & a \ $ \ a \ b \ a \ [\text{a}_2] \\
b_1 & a \ a \ b \ a \ $ \ [\text{a}_3] \\
\end{array} \\
\begin{array}{ll}
F & L \\
\$ & a \ b \ a \ a \ b \ a_0 \\
a_0 & $ \ a \ b \ a \ a \ b_0 \\
a_1 & a \ b \ a \ $ \ a \ b_1 \\
a_2 & b \ a \ $ \ a \ b \ a_1 \\
a_3 & b \ a \ a \ b \ a \ $ \\
b_0 & a \ $ \ a \ b \ a \ a_2 \\
b_1 & a \ a \ b \ a \ $ \ a_3 \\
\end{array}
\]

get\_frange(‘a’, 2, 3)\rightarrow[3,4]

SA[3] = 3, SA[4] = 0 \rightarrow \text{Return} \ {0, 3}$
Finding all matches of $P$ occurs in $T$ in FM Index is ___________ time
Assignment 9: a_fmi

Learning Objective:

Construct a full FM Index

Implement exact pattern matching on a FM Index

Consider: How would you modify the provided code to handle subsampling in the Occurrence Table (OT) or Suffix Array (SA)?
**FM Index**

Let $a =$ fraction of rows we keep

Let $b =$ fraction of SA elements we keep

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>482</td>
<td>432</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SA'</th>
</tr>
</thead>
<tbody>
<tr>
<td>![SA']</td>
</tr>
</tbody>
</table>

FM Index consists of these, plus $L$ and $F$ columns

Note: suffix tree/array didn't have parameters like $a$ and $b$
FM Index

Components of FM Index:  
(blue indicates what we can adjust by changing $a$ & $b$)

First column ($F$): $\sim |\Sigma|$ integers  
Last column ($L$): $m$ characters

SA sample: $m \cdot a$ integers, $a$ is fraction of SA elements kept
OT Checkpoints: $m \cdot |\Sigma| \cdot b$ integers, $b$ is fraction of tallies kept

For DNA alphabet (2 bits / nt), $T =$ human genome, $a = 1/32$, $b = 1/128$:

First column ($F$): 16 bytes  
Last column ($L$): 2 bits * 3 billion chars = 750 MB  
SA sample: 3 billion chars * 4 bytes / 32 = ~ 400 MB  
OT Checkpoints: 3 billion * 4 alphabet chars * 4 bytes / 128 = ~ 400 MB

Total $\approx$ 1.5 GB  
~0.5 bytes per input char
FM Index: Small Memory Footprint

Suffix tree

\[ \geq 45 \text{ GB} \]

Suffix array

\[ \geq 12 \text{ GB} \]

FM Index

\[ \sim 1.5 \text{ GB} \]
## Suffix-Based Index Bounds

<table>
<thead>
<tr>
<th></th>
<th>Suffix tree</th>
<th>Suffix array</th>
<th>FM Index</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time: Does $P$ occur?</strong></td>
<td>$O(n)$</td>
<td>$O(n \log m)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td><strong>Time: Count $k$ occurrences of $P$</strong></td>
<td>$O(n + k)$</td>
<td>$O(n \log m)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td><strong>Time: Report $k$ locations of $P$</strong></td>
<td>$O(n + k)$</td>
<td>$O(n \log m + k)$</td>
<td>$O(n + k)$</td>
</tr>
<tr>
<td><strong>Space</strong></td>
<td>$O(m)$</td>
<td>$O(m)$</td>
<td>$O(m)$</td>
</tr>
<tr>
<td><strong>Needs $T$?</strong></td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td><strong>Bytes per input character</strong></td>
<td>$&gt;15$</td>
<td>$\sim 4$</td>
<td>$\sim 0.5$</td>
</tr>
</tbody>
</table>

$m = |T|$, $n = |P|$, $k = \#$ occurrences of $P$ in $T$