BWT for repetitive texts, part 3: Toehold lemma setup
Ben Langmead
## Locate queries

<table>
<thead>
<tr>
<th></th>
<th>Count</th>
<th>Locate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Space</td>
<td>Time</td>
</tr>
<tr>
<td>FM Index (2000)</td>
<td>$O(n)$</td>
<td>$O(m)$</td>
</tr>
<tr>
<td>RLFM Index (2005)</td>
<td>$O(r)$</td>
<td>$O(m)$</td>
</tr>
<tr>
<td>r-index (2018)</td>
<td>$O(r)$</td>
<td>$O(m)$</td>
</tr>
</tbody>
</table>

$n = \text{reference length, } m = \text{query length,}$

$r = \# \text{BWT runs}$

(bounds simplified)
<table>
<thead>
<tr>
<th></th>
<th>Count</th>
<th>Locate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Space</td>
<td>Time</td>
</tr>
<tr>
<td>FM Index (2000)</td>
<td>$O(n)$</td>
<td>$O(m)$</td>
</tr>
<tr>
<td>RLFM Index (2005)</td>
<td>$O(r)$</td>
<td>$O(m)$</td>
</tr>
<tr>
<td>r-index (2018)</td>
<td>$O(r)$</td>
<td>$O(m)$</td>
</tr>
</tbody>
</table>

$n =$ reference length, $m =$ query length, $r =$ # BWT runs

How?

Locate query & SA samples

A precomputed “suffix array” stores the answers...

$$
\begin{align*}
F & \quad L \\
$ & a b a a b a \\
a & a b a a b a \\
a & a b a a b \\
\textcolor{red}{a} & \textcolor{red}{b} a a b a \textcolor{red}{a} \\
\textcolor{red}{a} & \textcolor{red}{b} a a b a \textcolor{red}{a} \\
b & a a b a \textcolor{red}{a} \\
b & a a b a \textcolor{red}{a} \\
\end{align*}
$$
A precomputed “suffix array” stores the answers…

<table>
<thead>
<tr>
<th>F</th>
<th>L</th>
<th>SA</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>5 a $</td>
<td></td>
</tr>
<tr>
<td>a a b a $ a b</td>
<td>2 a a b a $</td>
<td></td>
</tr>
<tr>
<td><strong>a b a $ a b a</strong></td>
<td>3 a b a $</td>
<td></td>
</tr>
<tr>
<td>a b a $ a b a</td>
<td>0 a b a a b a $</td>
<td></td>
</tr>
<tr>
<td>b a $ a b a a</td>
<td>4 b a $</td>
<td></td>
</tr>
<tr>
<td>b a a b a $ a</td>
<td>1 b a a b a $</td>
<td></td>
</tr>
</tbody>
</table>
A precomputed “suffix array” stores the answers…

Locate query & SA samples

$n$ integers is too big, so we sample instead
Locate query & SA samples

$ a b a a b a$
$a b a a b a$
$a a b a a b$
$ a b a $ a b$
$b a a b a a$
$b a a a b a$

SSA (evens only)

Offset: 0

6
2
0
4
Locate query & SA samples

$ a b a a b a$
$a b a b a a b$
$a b a a b a$
$b a b a b a$
F

L

SSA (evens only)

Offset: ?
Locate query & SA samples

$$\begin{align*}
F & \quad L \\
\$ & a \ b \ a \ b \ a \\
a & a \ b \ a \ a \ b \\
a & a \ b \ a \$ \ a \ b \\
a & b \ a \ a \ b \ a \\
b & a \ a \ b \ a \$ \\
a \ b \ a \ a \ b \ a \ b \\
a & b \ a \ a \ b \ a \\
\end{align*}$$

Offset: 2
Offset: ?

SSA (evens only)

$$\begin{align*}
6 & \\
2 & \\
0 & \\
4 & \\
\end{align*}$$
Locate query & SA samples

\[ F \]

\[ L \]

\[ \text{Offset: 2} \]

\[ \text{Offset: 3} \]

SSA (evens only)

\[ \begin{array}{c}
6 \\
2 \\
0 \\
4 \\
\end{array} \]
Locate query & SA samples

$$
\begin{align*}
F & \quad L \\
\$ & a & b & a & b & a \\
a & a & b & a & a & b \\
a & a & b & a & b & a \\
b & a & b & a & b & a \\
b & a & a & b & a & a \\
\end{align*}
$$

SSA (multiples of 4 only)

\[ \begin{array}{c}
0 \\
4 \\
\end{array} \]

Offset: ?
Locate query & SA samples

\[ F \quad L \]

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

SSA (multiples of 4 only)

\[
\begin{array}{c}
0 \\
4 \\
\end{array}
\]

Offset: 0

Offset: ?
Locate query & SA samples

\[ F \quad \begin{array}{lllllll}
\$ & a & b & a & a & b & a \\
a & b & a & b & a & a & b \\
a & a & b & a & b & a & b \\
b & a & a & b & a & b & a \\
b & a & a & b & a & b & a \\
\end{array} \]

\[ L \quad \begin{array}{lllllll}
a & b & a & a & b & a \\
a & b & a & b & a & b \\
a & b & a & b & a & b \\
b & a & a & b & a & b \\
b & a & a & b & a & b \\
\end{array} \]

SSA (multiples of 4 only)

\[
\begin{array}{c}
0 \\
4 \\
\end{array}
\]

Offset: 3
Offset: 0
Locate query & SA samples

<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th>L</th>
<th>SSA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$</td>
<td>a b a a b</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>$ a b a a b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>a b a $ a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b a a b a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>a b a a b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>a a b a $</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If we store SAs for multiples of constant \(c\), max LF steps is \(c\), but SSA sample takes \(O(n/c) = O(n)\) space

Or let \(c\) be a function of \(n\), like \(\sqrt{n} \rightarrow O(\sqrt{n})\) space & time

For repetitive texts, \(O(r)\) space is much better, if queries can be made efficient...
Fundamental movements

$T$ space $\leftrightarrow$ BWM($T$) $\leftrightarrow$ Lex space

$SA \quad SA^{-1}$
Fundamental movements: LF

$T \xrightarrow{SA} BWM(T) \xrightarrow{\text{BWT}(T)}$

$q' \leftarrow \text{LF}[q]$  
$q'' \leftarrow \text{LF}[q']$  
$q''' \leftarrow \text{LF}[q'']$

Start here
An LF step (in lex space)...

\[ \text{SA}[\text{LF}[q]] = \text{SA}[q] - 1 \]

...moves to the left by 1 in T space
Fundamental movements: LF

An LF step (in lex space)...

\[ SA[\text{LF}[q]] = SA[q] - 1 \]

...moves to the left by 1 in T space
Fundamental movements: $\phi$

$$SA^{-1}[\phi(i)] = SA^{-1}[i] - 1$$
Fundamental movements: "Runny LF"

When $T$ is repetitive, LF can be thought of as permuting **runs**

I.e. LF mapping is "runny"
Fundamental movements: "Runny LF"

When $T$ is repetitive, LF can be thought of as permuting **runs**

I.e. LF mapping is "runny"
Runny LF

L  trrrwwwwwwwwwooo__bbbyyyyyyyrrrrrrruuutt$____aaaao0000000000

F  $________aaaabbboooooooooooooooooorrrrrrrrrrrrrrrrutuuuwwwwwy
Runny LF principle

If $BWT[q] = BWT[q + 1]$, then $LF[q + 1] = LF[q] + 1$

Two positions in the same BWT run move through LF in a parallel fashion
O(r) locate queries

Algorithm has two components:

1. We always have a "toehold"

2. We can get the rest from the toehold

O(r) locate queries

Algorithm has two components:

1. We always have a "toehold"

2. We can get the rest from the toehold

Locate query & SA samples

Samples at run heads
Locate query & SA samples

Samples at run heads
Locate query & SA samples

Samples at run heads
Locate query & SA samples

$T$ row_row_row_your_boat row_row_row_your_boat row_row_row_your_boat$

$L$ trrrwwwwwwwooo___bbbyyrrrrrrruuutt$______aaaoooooooooooo___

Samples at run heads

62 57 44
Locate query & SA samples

T  row_row_row_your_boatrow_row_row_your_boatrow_row_row_your_boat$

L  trrrwwwwwwwwwooo__bbyyyrrrrrrruutt$____aao000000000

Samples at run heads
Locate query & SA samples

$T$
row_row_row_your_boatrow_row_row_your_boatrow_row_row_your_boat$

$L$

Samples at run heads
62 57 44 60 58 59 54 42 56 41 63 45 61 55 53

Samples at run tails
62 15 10 18 16 17 12 8 14 20 63 7 19 9 11
Locate query & SA samples

Samples at run heads & tails
Standard backward search

Usually we do backward search first, locate queries second:
Backward search with toehold

Augment backward search so that at every step, we have one offset $j$ in our range $[sp, ep]$ for which we know $SA[j]$.

How to update $j$ and $SA[j]$ at each step?
Backward search with toehold

Say we have a $j$, $[sp, ep]$ and $SA[j]$ for step $i$ and we're updating for step $i + 1$. Next query character is $y$.

Case 1: $BWT[j^{(i)}] = y$

Case 2: $BWT[j^{(i)}] \neq y$

Superscript $(i)$ denotes value at step $i$
Backward search with toehold

Case 1: $BWT[j^{(i)}] = y$

LF does all the work
Backward search with toehold

Case 1: $BWT[j^{(i)}] = y$

$$LF[j^{(i)}]$$

$$j^{(i+1)} = LF[j^{(i)}]$$

$$SA[j]^{(i+1)} = SA[j]^{(i)} - 1$$

LF does all the work
Backward search with toehold

Case 2: $BWT[j^{(i)}] \neq y$

...xyz...

Case 2a: **No** element in $BWT[sp^{(i)} \ldots ep^{(i)}]$ equals $y$

No matches found, no locate queries needed

Case 2b (next slide): Some elements in $BWT[sp^{(i)} \ldots ep^{(i)}]$ match $y$, but $BWT[j^{(i)}]$ doesn't
Backward search with toehold

Case 2b: Some elements in $\text{BWT}[sp^{(i)} \ldots ep^{(i)}]$ match $y$, but $\text{BWT}[j^{(i)}]$ doesn't.

$\ldots xy z \ldots$

$sp^{(i)}$

$ep^{(i)}$

BWT$[sp^{(i)} \ldots ep^{(i)}]$ must contain a $y$-run head or tail.
Case 2b: Some elements in $\text{BWT}[sp^{(i)} \ldots ep^{(i)}]$ match $y$, but $\text{BWT}[j^{(i)}]$ doesn't.

$\text{BWT}[sp^{(i)} \ldots ep^{(i)}]$ \textbf{must} contain a $y$-run head or tail.
Backward search with toehold

**Case 2b:** Some elements in $\text{BWT}[sp^{(i)} \ldots ep^{(i)}]$ match $y$, but $\text{BWT}[j^{(i)}]$ doesn't

Let $j^{(i+1)}$ be the offset of a $y$-head or $y$-tail. Look up $\text{SA}[j^{(i+1)}]$ in our structure holding SA items at heads/tails.

$\text{BWT}[sp^{(i)} \ldots ep^{(i)}]$ **must** contain a $y$-run head or tail.
Toehold predecessor structure

How exactly to look up $SA[j^{(i+1)}]$ in our structure holding SA items at heads/tails?

Recall our discussion of the **predecessor structure** from the last video...
Run-length FM Index

\[ B \cdot \text{rank}_1 \]
\[ B \cdot \text{select}_1 \]

\[ [0, m) \]

\text{rank}_1 \text{ answered with binary search on } S_1

\text{select}_1 \text{ is a direct lookup in } S_1

Associating values with set bits is as easy as adding \textbf{more arrays} parallel to \( S_1 \).
Toehold predecessor structure

Similarly, we can add associativity to the structure we referenced here:

<table>
<thead>
<tr>
<th>Run-length FM Index</th>
<th>RLFM index:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Predecessor queries on</strong> (B)</td>
<td>(r \leftarrow B \cdot \text{rank}_1(\text{off} + 1))</td>
</tr>
<tr>
<td>(O(\log \log n/r)) time, (O(r)) space</td>
<td>(o \leftarrow B \cdot \text{select}_1(r))</td>
</tr>
<tr>
<td>Belazzougui &amp; Navarro, Thm A.1</td>
<td>(pv \leftarrow S \cdot \text{rank}_c(r - 1))</td>
</tr>
</tbody>
</table>

| **Rank queries on** \(S\) | \(s_1 \leftarrow C[c]\) |
| \(O(\log \log \sigma)\) time, \(O(r)\) space | \(s_2 \leftarrow B'[c] \cdot \text{select}_1(pv)\) |
| Belazzougui & Navarro, Thm 5.4 | \(\text{next\_off} \leftarrow s_1 + s_2 + (\text{off} - o)\) |

| **Select queries on** \(C \& B'\) | Array of cumulative counts |
| \(O(1)\) time, \(O(r)\) space | |


Toehold predecessor structure

Final point: structure needs to be stratified by character

Increases time to $O(\log \log (\sigma + n/r))$
Toehold predecessor structure

\[
\begin{array}{c|c|c}
\text{\(R_a\)} & \text{SSA}_a & \text{\(R_b\)} \\
1 & 103 & 1 \\
1 & 290 & 1 \\
1 & 202 & 1 \\
\end{array}
\]
We always have a "toehold"

We can get the rest from the toehold