String Algorithms and Data Structures

FM Index

CS 199-225
Brad Solomon

March 28, 2022
A_sarray reflection

Only one response :(
A_bwt due today!

BWT is a key part of today’s lecture as well
Submit suggestions for end-of-semester material!
Burrows-Wheeler Transform

Reversible permutation of the characters of a string

$ a b a a b a $  

$ b a a a b a $  

BWT(T)
Burrows-Wheeler Transform: 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).

Any ranking we give to characters in $T$ will match in $F$ and $L$. 

They're sorted by right-context.
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: \( F = \$\text{CCCGGG} \)

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

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

BWT(T) = e$ppp$a

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:

$L$ 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

\[ p = A \ A \ A \]

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

\[ P = B \ A \ B \]

\[
\begin{align*}
&\$$ B B B A A A_0 \\
&A_0 A_0 B B B A A_1 \\
&A_1 A A_1 A B B A_2 \\
&A_2 A A_2 A B B_0 \\
&B_0 B_0 A A A A B B B_1 \\
&B_1 B_1 B A A A A B B B_2 \\
&B_2 B_2 B B A A A A B$
\end{align*}
\]
(1) Scanning for preceding character in $L$ is slow

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

We don’t store ranks!

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?

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 # \texttt{a}s, \texttt{b}s in \(L\) up to every row:

\[
\begin{array}{cccc}
L & a & b \\
\hline
\texttt{a} & & \\
\texttt{b} & & \\
\texttt{b} & & \\
\texttt{a} & & \\
\$ & & \\
\texttt{a} & & \\
\texttt{a} & & \\
\end{array}
\]
**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:

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

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

What two indices should I look up for $P = bb$? What ranks did we find?

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

Table is $m \times \mid \Sigma \mid$ integers — *that’s worse than a suffix array!*
Next idea: pre-calculate # 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>$</td>
<td>a</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>$</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>

Checkpoint 1

Checkpoint 2
Next idea: pre-calculate \# `a`s, `b`s in `L` up to some rows, e.g. every 5\textsuperscript{th} row. Call pre-calculated rows *checkpoints*.

\begin{array}{|c|c|c|c|}
\hline
F & L & a & b \\
\hline
\$ & a & 1 & 0 \\
\hline
a & b & & \\
\hline
a & b & & \\
\hline
a & a & & \\
\hline
a & \$ & & \\
\hline
b & a & 3 & 2 \\
\hline
b & a & & \\
\hline
\end{array}
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>$$</td>
<td>a</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>$$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>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 $O(1)$.
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...

\[ P = \text{aba} \]

Got the same range, \([3, 4]\), we would have got from suffix array

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>$</td>
<td>a</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>$</td>
<td>a</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>a</td>
<td>$</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>a</td>
<td>$</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>$</td>
<td>a</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>$</td>
<td>a</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>$</td>
<td>a</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>$</td>
<td>a</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>$</td>
<td>a</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>$</td>
<td>a</td>
<td>b</td>
<td>a</td>
</tr>
</tbody>
</table>

Where are these?
FM Index: Suffix Array Sampling

Idea: store some suffix array elements, but not all

```
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 $ a b a a b
a a b a $ a b
```

```
```

Lookup for row 4 succeeds

Lookup for row 3 fails - SA entry was discarded
FM Index: Suffix Array Sampling

LF Mapping tells us that “a” at the end of row 3 corresponds to...

...“a” at the beginning of row 2

<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>2</td>
<td></td>
</tr>
<tr>
<td>a b a $ a b a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a b a a b a $</td>
<td>0</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>

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:

\[
\begin{array}{ccc}
F & L & SA' (every 4th) \\
$ & a & b & a & a & b & a & \quad & 0 \\
a & a & b & a & a & b & & \\
a & a & b & a & a & b & & \\
a & a & b & a & a & b & & \\
a & b & a & a & b & a & & \\
\end{array}
\]

Starting here →
FM Index: Suffix Array Sampling

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

Starting here
FM Index: Suffix Array Sampling

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

Starting here... →

```
<table>
<thead>
<tr>
<th>F</th>
<th>L</th>
<th>SA' (every 4th)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>$a</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>$a</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>$a</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>$a</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>$a</td>
<td></td>
</tr>
</tbody>
</table>
```

Starting here
FM Index: Suffix Array Sampling

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

Starting here ...

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

\[
\begin{align*}
\$ & a b a a b a \\
 a & $ a b a a b \\
 a & a b a $ a b \\
 a & b a $ a b a \\
 a & b a a b a $ \\
 b & a $ a b a a \\
 b & a a b a $ a
\end{align*}
\]

Three steps

Index: 0

Index: 0 + 3 = 3

*Lets put all these pieces together…*
FM Index: Querying

\[ P = \texttt{aba} \]

\[
\begin{array}{cccccc}
  & F & & L & \\
\$ & a & b & a & a & b & a_0 \\
  a_0 & $ & a & b & a & a & b \\
a_1 & a & b & a & $ & a & b \\
a_2 & b & a & $ & a & b & a_1 \\
a_3 & b & a & a & b & a & $ \\
  b & a & $ & a & b & a & a_2 \\
  b & a & a & b & a & $ & a_3 \\
\end{array}
\]

\texttt{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

\( P = \text{aba} \)

\[
\begin{align*}
F & \quad L \\
\$ & \quad a \ b \ a \ a \ b \ a_0 \\
a_0 & \quad a \ b \ a \ a \ b \ b \ a_1 \\
a_1 & \quad a \ b \ a \ $ \ a \ b \ b \ a_2 \\
a_2 & \quad b \ a \ a \ b \ a \ $ \\
a_3 & \quad b \ a \ a \ b \ a \ $ \\
b & \quad b \ a \ $ \ a \ b \ a \ b \ a_2 \\
b & \quad b \ a \ a \ b \ a \ $ \ a_3 \\
\end{align*}
\]

get\_frange() \quad get\_lrange()
```cpp
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

\[ P = \text{aba} \]

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

get_frange() \quad \Rightarrow \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

$p = \text{aba}$

\[
\begin{array}{c|c}
 F & L \\
 \hline
 \$ & a b a a b a_0 \\
a_0 & a b a a b \\
a_1 & a b a \$ a b \\
a_2 & b a \$ a b a_1 \\
a_3 & b a a b a \$ \\
b & a \$ a b a_2 \\
b & a a b a \$ a_3 \\
\end{array}
\]
$P = \text{aba}$

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

get_lrange(‘a’,5,6)->[2,4]

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

get_frange(‘a’,2,3)->[3,4]

SA[3] = 3, SA[4] = 0 --> Return {0, 3}
**FM Index**

$| T | = m, \ | P \ | = n$

<table>
<thead>
<tr>
<th>$P = \text{aba}$</th>
<th>$P = \text{aba}$</th>
<th>$P = \text{aba}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F$</td>
<td>$L$</td>
<td>$F$</td>
</tr>
<tr>
<td>$$ a b a a b $a_0$</td>
<td>$$ a b a a b $a_0$</td>
<td>$$ a b a a b $a_0$</td>
</tr>
<tr>
<td>$a_0$ a b a a a b</td>
<td>$a_0$ a b a a a b</td>
<td>$a_0$ a b a a a b</td>
</tr>
<tr>
<td>$a_1$ a b a $a b$</td>
<td>$a_1$ a b a $a b$</td>
<td>$a_1$ a b a $a b$</td>
</tr>
<tr>
<td>$a_2$ b a $a b a b$</td>
<td>$a_2$ b a $a b a b$</td>
<td>$a_2$ b a $a b a b$</td>
</tr>
<tr>
<td>$a_3$ b a b a $a$</td>
<td>$a_3$ b a b a $a$</td>
<td>$a_3$ b a b a $a$</td>
</tr>
<tr>
<td>$b$ a $a b a a$</td>
<td>$b$ a $a b a a$</td>
<td>$b$ a $a b a a$</td>
</tr>
<tr>
<td>$b$ a a b a $a_3$</td>
<td>$b$ a a b a $a_3$</td>
<td>$b$ a a b a $a_3$</td>
</tr>
</tbody>
</table>

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 sub-sampling 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>
<th>SA'</th>
</tr>
</thead>
<tbody>
<tr>
<td>482</td>
<td>432</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>44</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>488</td>
<td>439</td>
<td></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
≥ 45 GB

Suffix array
≥ 12 GB

FM Index
~ 1.5 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>Time: Does P occur?</td>
<td>$O(n)$</td>
<td>$O(n \log m)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Time: Count $k$</td>
<td>$O(n + k)$</td>
<td>$O(n \log m)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>occurrences of P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time: Report $k$</td>
<td>$O(n + k)$</td>
<td>$O(n \log m + k)$</td>
<td>$O(n + k)$</td>
</tr>
<tr>
<td>locations of P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Space</td>
<td>$O(m)$</td>
<td>$O(m)$</td>
<td>$O(m)$</td>
</tr>
<tr>
<td>Needs T?</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Bytes per input</td>
<td>$&gt;15$</td>
<td>$\sim 4$</td>
<td>$\sim 0.5$</td>
</tr>
<tr>
<td>character</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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