Burrows-Wheeler Transformation

- Originally developed for data compression
- Reordering text -> Better locality = better compression
  - Used in bzip2
- Additional data structures for sequence search
  - Ferragina-Manzini index
  - “Summarized” suffix array
Burrows-Wheeler Transformation

1. Append to the input string a special char, $, smaller than all alphabet. $mississippi$
2. Generate all rotations.
Burrows-Wheeler Transformation (cnt’d)

3. Sort rotations according to the alphabetical order.
### Burrows-Wheeler Transformation (cnt’d)

4. Output the last column.

| $ | m | i | s | s | s | s | s | s | i | p | p | i |
|---|---|---|---|---|---|---|---|---|---|---|---|
| i | $ | m | i | s | s | s | s | s | i | p | p | $ |
| i | p | p | i | $ | m | i | s | s | s | s | s | s |
| i | s | s | i | p | p | i | $ | m | i | s | s | s |
| i | s | s | i | s | s | s | i | p | p | i | $ | m |
| m | i | s | s | s | i | s | s | i | p | p | i | $ |
| p | i | $ | m | i | s | s | s | s | i | s | s | i |
| p | p | i | $ | m | i | s | s | s | s | s | s | s |
| s | i | p | p | i | $ | m | i | s | s | s | s | s |
| s | i | s | s | i | p | p | i | $ | m | i |
| s | s | i | s | s | s | i | p | p | i | $ | m | i |
Burrows-Wheeler Transformation (cnt’d)

\[
\text{mississippi}$
\]

\[
\downarrow
\]

\[
ipssm$pissii$
\]
BWT – alternative construction

$T = a \ b \ a \ a \ b \ a$

**BWT**

$\$

$ a \ b \ a \ a \ b$

$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 \$ $a$

**Suffix Array**

6 $$

5 a $$

2 $a \ a \ b \ a \$

3 $a \ b \ a \$

0 $a \ b \ a \ a \ b \ a \$

4 $b \ a \$

1 $b \ a \ a \ b \ a \$

$\$\$\$

$BWT[i] = \begin{cases} 
T[SA[i] - 1], & \text{if } SA[i] > 0 \\
\$, & \text{if } SA[i] = 0 
\end{cases}$

$BWT = \text{characters just to the left of characters in } SA$
L to F map

Let's make an L to F map.

Observation:
The $n^{\text{th}}$ i in L is the $n^{\text{th}}$ i in F.
L to F map

Store/compute a two dimensional Occ\((j, 'c')\) table of the number of occurrences of char ‘c’ up to position \(j\) (inclusive).

and one dimensional Cnt(‘c’) and Rank(‘c’) tables

<table>
<thead>
<tr>
<th>(i)</th>
<th>(m)</th>
<th>(p)</th>
<th>(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(\text{Cnt('c')})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{Rank('c')})</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(\text{Occ}(j, 'c'))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{Cnt('c')})</td>
</tr>
<tr>
<td>(\text{Rank('c')})</td>
</tr>
</tbody>
</table>
L to F map

\[ \text{Cnt}('\$') + \text{Cnt('i')} + \text{Cnt('m')} + \text{Cnt('p')} = 8 \]

\[ + [\text{Occ}(9, 's') = 3] \]

\[ = 11 \]
L to F map

(1) i
(2) p
(3) s
(4) s
(5) m
(6) $
(7) p
(8) i
(9) s
(10) s
(11) i
(12) i
Search with BWT-FM: L to F map

### Original sequence
- $\text{gcagcagact}$

### BWT
- $\text{t}$
- $\text{g}$
- $\text{c}$
- $\text{c}$
- $\text{g}$

### SA
- **1**: 9
  - **BWT**: $\text{act}$$\text{agcagcag}$
- **2**: 7
  - **BWT**: $\text{agact}$$\text{agcagc}$
- **3**: 4
  - **BWT**: $\text{agcagact}$$\text{agc}$
- **4**: 1
  - **BWT**: $\text{agcagcagact}$$\text{agc}$
- **5**: 6
  - **BWT**: $\text{cagact}$$\text{agcag}$
- **6**: 3
  - **BWT**: $\text{cagcagact}$$\text{ag}$
- **7**: 10
  - **BWT**: $\text{ct}$$\text{agcagcaga}$
- **8**: 8
  - **BWT**: $\text{gact}$$\text{agcagca}$
- **9**: 5
  - **BWT**: $\text{gcagact}$$\text{agc}$
- **10**: 2
  - **BWT**: $\text{gcagcagact}$$\text{ag}$
- **11**: 11
  - **BWT**: $\text{t}$$\text{agcagcagac}$
Search with BWT-FM: FM-index

Auxiliary data structures for efficient pattern matching: how to find the corresponding chars in the first column efficiently, in terms of both time and space.

Original sequence

<table>
<thead>
<tr>
<th>SA</th>
<th>$agcagcagact</th>
<th>BWT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
<td>t</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>g</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>c</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>$</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>g</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>g</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>a</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>a</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>a</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>c</td>
</tr>
</tbody>
</table>

FM indices

<table>
<thead>
<tr>
<th>rank</th>
<th>a</th>
<th>c</th>
<th>g</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>8</td>
<td>11</td>
<td></td>
</tr>
</tbody>
</table>
Search with BWT-FM: FM-index

Auxiliary data structures for efficient pattern matching: how to find the corresponding chars in the first column efficiently, in terms of both time and space.

Original sequence

BWT

FM indices

Next block: From 1 + 0 = 1 to 1 + (4-1) = 4
Search with BWT-FM: FM-index

Auxillary data structures for efficient pattern matching: how to find the corresponding chars in the first column efficiently, in terms of both time and space.

Original sequence

<table>
<thead>
<tr>
<th>SA</th>
<th>$agcagcagact</th>
<th>BWT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>g</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>t</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>$</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>c</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>g</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>a</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>g</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>c</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>t</td>
</tr>
</tbody>
</table>

Next block: From 5 + 0 = 5 to 5 + (2-1) = 6

<table>
<thead>
<tr>
<th>rank</th>
<th>a</th>
<th>c</th>
<th>g</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>8</td>
<td>11</td>
<td></td>
</tr>
</tbody>
</table>

FM indices
Auxiliary data structures for efficient pattern matching: how to find the corresponding chars in the first column efficiently, in terms of both time and space.

**Original sequence**

**BWT**

**FM indices**

Next block: From 8 - 1 = 9 to 8 + (3-1) = 10
Inexact match
Videos

- BWT
  - [https://www.youtube.com/watch?v=4n7NPk5lwbl](https://www.youtube.com/watch?v=4n7NPk5lwbl)

- FM-index
  - [https://www.youtube.com/watch?v=kvVGj5V65io](https://www.youtube.com/watch?v=kvVGj5V65io)