Recap: FM-Index and r-Index

- based on backwards-search
- used to answer rank-queries on BWT

Function \( \text{BackwardsSearch}(P[1..n], C, \text{rank}) \):

1. \( s = 1, e = n \)
2. \( \text{for } i = m, \ldots, 1 \text{ do} \)
3. \( s = C[P[i]] + \text{rank}_{P[i]}(s - 1) + 1 \)
4. \( e = C[P[i]] + \text{rank}_{P[i]}(e) \)
5. \( \text{if } s > e \text{ then} \)
6. \( \quad \text{return } \emptyset \)
7. \( \text{return } [s, e] \)
Recap: FM-Index and $r$-Index

- based on backwards-search
- used to answer rank-queries on BWT

**FM-Index**
- build wavelet tree directly on BWT
- wavelet tree can be $H_0$ compressed
- blind to repetitions

**Function** BackwardsSearch($P[1..n]$, $C$, rank):

1. $s = 1$, $e = n$
2. for $i = m, \ldots, 1$ do
3.     $s = C[P[i]] + rank_{P[i]}(s - 1) + 1$
4.     $e = C[P[i]] + rank_{P[i]}(e)$
5.     if $s > e$ then
6.         return $\emptyset$
7.     return $[s, e]$
Recap: FM-Index and $r$-Index

- based on backwards-search
- used to answer rank-queries on BWT

**FM-Index**
- build wavelet tree directly on BWT
- wavelet tree can be $H_0$ compressed
- blind to repetitions

**r-Index**
- many arrays with $r$ entries
- build wavelet tree on one of these arrays
- size in numbers of BWT runs $r$

**Function** $\text{BackwardsSearch}(P[1..n], C, \text{rank})$:  
1. $s = 1$, $e = n$
2. for $i = m, \ldots, 1$ do
3. \hspace{1em} $s = C[P[i]] + \text{rank}_{P[i]}(s - 1) + 1$
4. \hspace{1em} $e = C[P[i]] + \text{rank}_{P[i]}(e)$
5. if $s > e$ then
6. \hspace{2em} return $\emptyset$
7. return $[s, e]$
Different Types of Compression

Statistical Coding

- based on frequencies of characters
- results in size $|T| \cdot H_k(T)$
  - $k$-th order empirical entropy
- good if frequencies are skewed
- blind to repetitions

$$\ell |T| \cdot H_k(T) \approx H_k(T)$$

LZ-Compression references to previous occurrences each LZ factor can be encoded in $O(1)$ space good for repetitions index in this lecture

BWT-Compression used in powerful index theoretical insight in this lecture
Different Types of Compression

### Statistical Coding
- based on frequencies of characters
- results in size $|T| \cdot H_k(T)$
- $k$-th order empirical entropy
- good if frequencies are skewed
- blind to repetitions
  \[
  |T \ldots T| \cdot H_k(T \ldots T) \approx \ell |T| \cdot H_k(T)
  \]

### LZ-Compression
- references to previous occurrences
- each LZ factor can be encoded in $O(1)$ space
- good for repetitions
- index in this lecture

LZ-Compression references to previous occurrences each LZ factor can be encoded in $O(1)$ space good for repetitions index in this lecture
## Different Types of Compression

### Statistical Coding
- Based on frequencies of characters
- Results in size $|T| \cdot H_k(T)$
- $k$-th order empirical entropy
- Good if frequencies are skewed
- Blind to repetitions

$$|T \ldots T| \cdot H_k(T \ldots T) \approx \ell |T| \cdot H_k(T)$$

### LZ-Compression
- References to previous occurrences
- Each LZ factor can be encoded in $O(1)$ space
- Good for repetitions
- Index in this lecture

### BWT-Compression
- Used in powerful index
- Theoretical insight in this lecture
Definition: LZ77 Factorization [ZL77]

Given a text \( T \) of length \( n \) over an alphabet \( \Sigma \), the **LZ77 factorization** is

- a set of \( z \) factors \( f_1, f_2, \ldots, f_z \in \Sigma^+ \), such that
- \( T = f_1 f_2 \ldots f_z \) and for all \( i \in [1, z] \) \( f_i \) is
- single character not occurring in \( f_1 \ldots f_{i-1} \) or
- longest substring occurring \( \geq 2 \) times in \( f_1 \ldots f_i \)

\[
T = abababbbaba$

\[f_1 = a\]
\[f_2 = b\]
\[f_3 = abab\]
\[f_4 = bbb\]
\[f_5 = aba\]
\[f_6 = $\]
LZ-Compressed Index

Definition: LZ77 Factorization [ZL77]

Given a text $T$ of length $n$ over an alphabet $\Sigma$, the
LZ77 factorization is
- a set of $z$ factors $f_1, f_2, \ldots, f_z \in \Sigma^+$, such that
- $T = f_1 f_2 \ldots f_z$ and for all $i \in [1, z]$ $f_i$ is
  - single character not occurring in $f_1 \ldots f_{i-1}$ or
  - longest substring occurring $\geq 2$ times in $f_1 \ldots f_i$

$LZ = abababbbaba$

- $f_1 = a$
- $f_2 = b$
- $f_3 = abab$
- $f_4 = bbb$
- $f_5 = aba$
- $f_6 = $
**Block Trees [Bel+21] (1/4)**

**Definition: Block Tree (1/4)**

Given a text $T$ of length $n$ over an alphabet of size $\sigma$
- $\tau, s \in \mathbb{N}$ greater 1
- assume that $n = s \cdot \tau^h$ for some $h \in \mathbb{N}$
  1. append $s$ until $n$ has this form

A block tree is a
- perfectly balanced tree with height $h$
- that may have leaves at higher levels such that
- the root has $s$ children,
- each other inner node has $\tau$ children
Definition: Block Tree (2/4)

In a block tree, leaves at
- the last level store characters or substrings of $T$
- at higher levels store special leftward pointer

Each node $u$
- represents a block $B^u$
- which is a substring of $T$ identified by a position

The root represents $T$ and its children consecutive blocks of $T$ of size $n/s$
Definition: Block Tree (3/4)

Let $\ell_u$ be the level (depth) of node $u$

- the level of the root is 0

Let $B_1, B_2, \ldots$ be the blocks represented at level $\ell_u$ from left to right

- for any $i$, $B_i$ and $B_{i+1}$ are consecutive in $T$
- if $B_i B_{i+1}$ are the leftmost occurrence in $T$, the nodes representing the blocks are marked
Definition: Block Tree (4/4)

If node \( u \) is marked, then
- it is an internal node
- with \( \tau \) children

otherwise, if node \( u \) is not marked, then
- \( u \) is a leaf storing
- pointers to nodes \( v_i, v_{i+1} \) at the same level
  - that represent blocks \( B_i \) and \( B_{i+1} \)
  - covering the leftmost occurrence of \( B^u \)
- offset to the occurrence of \( B^u \) in \( B_i B_{i+1} \)

leaves on last level store text explicitly
### Definition: Block Tree (4/4)

If node $u$ is marked, then
- it is an internal node
- with $\tau$ children

otherwise, if node $u$ is not marked, then
- $u$ is a leaf storing
- pointers to nodes $v_i, v_{i+1}$ at the same level
  - that represent blocks $B_i$ and $B_{i+1}$
  - covering the leftmost occurrence of $B^u$
- offset to the occurrence of $B^u$ in $B_iB_{i+1}$

leaves on last level store text explicitly

- $|B^u| = n / (s \tau^{\ell_u - 1})$
- if $|B_u|$ is small enough, store text explicitly
  - $|B^u| \in \Theta(\lg \sigma \cdot n)$
Definition: Block Tree (4/4)

If node $u$ is marked, then
- it is an internal node
- with $\tau$ children

otherwise, if node $u$ is not marked, then
- $u$ is a leaf storing
- pointers to nodes $v_i, v_{i+1}$ at the same level
  - that represent blocks $B_i$ and $B_{i+1}$
  - covering the leftmost occurrence of $B^u$
- offset to the occurrence of $B^u$ in $B_i B_{i+1}$

leaves on last level store text explicitly

\[
|B^u| = n/(s\tau^{\ell_u-1})
\]

if $|B_u|$ is small enough, store text explicitly

$|B_u| \in \Theta(\lg_\sigma n)$

PINGO how many blocks are there per level?
Lemma: Number of Blocks per Level

The number of blocks in any level $> 0$ in the block tree is at most $3\tau z$
Lemma: Number of Blocks per Level

The number of blocks in any level \( \ell > 0 \) in the block tree is at most \( 3 \tau z \).

Proof (Sketch)

Let \( \ell > 0 \) be a level in the block tree and

- \( C = B_{i-1}B_iB_{i+1} \) a concatenation of three consecutive blocks at level \( \ell - 1 \)
- not containing the end of an LZ factor
- thus a leftwards occurrence in \( T \)
Lemma: Number of Blocks per Level

The number of blocks in any level \( \ell > 0 \) in the block tree is at most \( 3\tau z \).

Proof (Sketch)

Let \( \ell > 0 \) be a level in the block tree and

- \( C = B_{i-1}B_iB_{i+1} \) a concatenation of three consecutive blocks at level \( \ell - 1 \)
- not containing the end of an LZ factor
- thus a leftwards occurrence in \( T \)

\( B_{i-1} \) and \( B_{i+1} \) can only be marked if \( B_i \) is marked

- \( B_i \) is marked if it contains end of LZ factor
- there are only \( z \) LZ factors
Lemma: Number of Blocks per Level
The number of blocks in any level \( \ell > 0 \) in the block tree is at most \( 3\tau z \)

Proof (Sketch)
Let \( \ell > 0 \) be a level in the block tree and
- \( C = B_{i-1}B_iB_{i+1} \) a concatenation of three consecutive blocks at level \( \ell - 1 \)
- not containing the end of an LZ factor
- thus a leftwards occurrence in \( T \)

\( B_{i-1} \) and \( B_{i+1} \) can only be marked if \( B_i \) is marked
- \( B_i \) is marked if it contains end of LZ factor
- there are only \( z \) LZ factors

Each marked block results in \( \tau \) children
Lemma: Number of Blocks per Level

The number of blocks in any level \( \ell > 0 \) in the block tree is at most \( 3\tau z \)

- \( O(\tau z) \) blocks per level

Proof (Sketch)

Let \( \ell > 0 \) be a level in the block tree and
- \( C = B_{i-1}B_iB_{i+1} \) a concatenation of three consecutive blocks at level \( \ell - 1 \)
- not containing the end of an LZ factor
- thus a leftwards occurrence in \( T \)

\( B_{i-1} \) and \( B_{i+1} \) can only be marked if \( B_i \) is marked
- \( B_i \) is marked if it contains end of LZ factor
- there are only \( z \) LZ factors

Each marked block results in \( \tau \) children
Lemma: Number of Blocks per Level

The number of blocks in any level $> 0$ in the block tree is at most $3\tau z$

- $O(\tau z)$ blocks per level
- unmarked block requires $O(\lg n)$ bits of space

Proof (Sketch)

Let $\ell > 0$ be a level in the block tree and

- $C = B_{i-1}B_iB_{i+1}$ a concatenation of three consecutive blocks at level $\ell - 1$
- not containing the end of an LZ factor
- thus a leftwards occurrence in $T$
- $B_{i-1}$ and $B_{i+1}$ can only be marked if $B_i$ is marked
- $B_i$ is marked if it contains end of LZ factor
- there are only $z$ LZ factors

Each marked block results in $\tau$ children
Lemma: Number of Blocks per Level

The number of blocks in any level $\ell > 0$ in the block tree is at most $3\tau z$

- $O(\tau z)$ blocks per level
- unmarked block requires $O(\lg n)$ bits of space
- marked block requires $O(\tau \lg n)$ bits of space

Proof (Sketch)

Let $\ell > 0$ be a level in the block tree and

- $C = B_{i-1}B_iB_{i+1}$ a concatenation of three consecutive blocks at level $\ell - 1$
- not containing the end of an LZ factor
- thus a leftwards occurrence in $T$

$B_{i-1}$ and $B_{i+1}$ can only be marked if $B_i$ is marked

- $B_i$ is marked if it contains end of LZ factor
- there are only $z$ LZ factors

Each marked block results in $\tau$ children
Lemma: Number of Blocks per Level

The number of blocks in any level > 0 in the block tree is at most $3\tau z$

- $O(\tau z)$ blocks per level
- unmarked block requires $O(\lg n)$ bits of space
- marked block requires $O(\tau \lg n)$ bits of space
- charged to child
- last level has $O(\tau z)$ blocks with plain text
  - $O(\lg_{\sigma} n)$ symbols of $\lfloor \lg n \rfloor$ bits
  - requiring $O(\lg \sigma)$ bits per block

Proof (Sketch)

Let $\ell > 0$ be a level in the block tree and
- $C = B_{i-1}B_iB_{i+1}$ a concatenation of three consecutive blocks at level $\ell - 1$
- not containing the end of an LZ factor
- thus a leftwards occurrence in $T$
- $B_{i-1}$ and $B_{i+1}$ can only be marked if $B_i$ is marked
- $B_i$ is marked if it contains end of LZ factor
- there are only $z$ LZ factors

Each marked block results in $\tau$ children
Lemma: Number of Blocks per Level

The number of blocks in any level \( \ell > 0 \) in the block tree is at most \( 3\tau z \)

- \( O(\tau z) \) blocks per level
- unmarked block requires \( O(\lg n) \) bits of space
- marked block requires \( O(\tau \lg n) \) bits of space
- charged to child
- last level has \( O(\tau z) \) blocks with plain text
  - \( O(\lg_\sigma n) \) symbols of \( \lfloor \lg n \rfloor \) bits
  - requiring \( O(\lg \sigma) \) bits per block
- \( h = \lg_\tau \frac{n \lg \sigma}{s \lg n} \) and \( O(s) \) pointers to top level

Proof (Sketch)

Let \( \ell > 0 \) be a level in the block tree and
- \( C = B_{i-1}B_iB_{i+1} \) a concatenation of three consecutive blocks at level \( \ell - 1 \)
- not containing the end of an LZ factor
- thus a leftwards occurrence in \( T \)

\( B_{i-1} \) and \( B_{i+1} \) can only be marked if \( B_i \) is marked
- \( B_i \) is marked if it contains end of LZ factor
- there are only \( z \) LZ factors

Each marked block results in \( \tau \) children
**Lemma: Number of Blocks per Level**

The number of blocks in any level \( \ell > 0 \) in the block tree is at most \( 3\tau z \)

- \( O(\tau z) \) blocks per level
- unmarked block requires \( O(\lg n) \) bits of space
- marked block requires \( O(\tau \lg n) \) bits of space
- charged to child
- last level has \( O(\tau z) \) blocks with plain text
  - \( O(\lg_{\sigma} n) \) symbols of \( \lfloor \lg n \rfloor \) bits
  - requiring \( O(\lg \sigma) \) bits per block
- \( h = \lg_{\tau} \frac{n \lg_{\sigma} \sigma}{s \lg n} \) and \( O(s) \) pointers to top level
- rounding up length adds \( \leq O(\tau) \) blocks per level

**Proof (Sketch)**

Let \( \ell > 0 \) be a level in the block tree and

- \( C = B_{i-1}B_iB_{i+1} \) a concatenation of three consecutive blocks at level \( \ell - 1 \)
- not containing the end of an LZ factor
- thus a leftwards occurrence in \( T \)

\( B_{i-1} \) and \( B_{i+1} \) can only be marked if \( B_i \) is marked

- \( B_i \) is marked if it contains end of LZ factor
- there are only \( z \) LZ factors

Each marked block results in \( \tau \) children
Lemma: Space Requirements of Block Trees

Given a text $T$ of length $n$ over an alphabet of size $\sigma$ and integers $s, \tau > 1$, a block tree of $T$ has height $h = \lg_{\tau} \frac{n \lg \sigma}{s \lg n}$. The block tree requires

$$O\left((s + z\tau \lg_{\tau} \frac{n \lg \sigma}{s \lg n}) \lg n\right)$$

bits of space,

where $z$ is the number of LZ77 factors of $T$. 

Block Trees are LZ Compressed (2/2)
Lemma: Space Requirements of Block Trees

Given a text $T$ of length $n$ over an alphabet of size $\sigma$ and integers $s, \tau > 1$, a block tree of $T$ has height

$$h = \lg_{\tau} \frac{n \lg \sigma}{s \lg n}.$$ 

The block tree requires

$$O((s + z \tau \lg_{\tau} \frac{n \lg \sigma}{s \lg n}) \lg n)$$

bits of space,

where $z$ is the number of LZ77 factors of $T$.

- $s = z$ results in a tree of height $O(\lg_{\tau} \frac{n \lg \sigma}{z \lg n})$.
- space requirements $O(z \tau \lg_{\tau} \frac{n \lg \sigma}{z \lg n} \lg n)$ bits
- however $z$ not known
Access Queries in Block Trees

- queries are easy to realize
- if not supported directly, additional information can be stored for blocks

Access Query

Given position $i$ return $T[i]$

- follow nodes that represent block containing $T[i]$
- of not marked follow pointer and consider offset
- at leaf, if last level, return character
- else, follow pointer and continue

- time $O(\lg \tau \frac{n}{s} \lg \sigma \frac{n}{s} \lg n)$
Access Queries in Block Trees

- Queries are easy to realize
- If not supported directly, additional information can be stored for blocks

**Access Query**

Given position $i$ return $T[i]$

- Follow nodes that represent block containing $T[i]$
- Of not marked follow pointer and consider offset
- At leaf, if last level, return character
- Else, follow pointer and continue

- Time $O(\lg \tau n \lg \sigma \frac{s}{\lg n})$

**Example on the board**

- PINGO: Can we answer rank queries the same way?
Rank Queries in Block Trees

- for each block add histogram $Hist_{Bu}$ for prefix of $T$ up to block (not containing)
- $O(\sigma(s + z\tau \lg n \frac{n \lg n}{s \lg n}) \lg n)$ bits of space

**Rank Query**

Given position $i$ and character $\alpha$ return $rank_\alpha(T, i)$

- follow nodes that represent block containing $T[i]$
- remember $Hist_{Bu}[\alpha]$
- of not marked follow pointer and consider offset
- at leaf, if last level, compute local rank for each character binary rank
- else, follow pointer and continue

- time $O(\lg \frac{n \log \sigma}{s \log n})$

- example on the board
Rank Queries in Block Trees

- for each block add histogram $Hist_{Bu}$ for prefix of $T$ up to block (not containing)
- $O(\sigma(s + zt \lg \frac{n \lg n}{s \lg \sigma}) \lg n)$ bits of space

Rank Query

Given position $i$ and character $\alpha$ return $\text{rank}_\alpha(T, i)$

- follow nodes that represent block containing $T[i]$
- remember $Hist_{Bu}[\alpha]$
- of not marked follow pointer and consider offset
- at leaf, if last level, compute local rank for each character binary
- else, follow pointer and continue

- time $O(\lg \frac{n \lg n}{s \lg \sigma})$

- example on the board

PINGO what can be problematic with block tree construction?
Construction of Block Trees

**O(n) Working Space**

- build Aho-Corasick automaton for containing all pairs of consecutive unmarked blocks
- identify unmarked blocks on next level
- \( O(n(1 + \lg \frac{z}{s})) \) time and \( O(n) \) space

Pruning

Size of block tree can be reduced further some blocks not necessary those blocks can easily be identified

\( O(s + \frac{z}{s}) \)

Working Space

replace Aho-Corasick automaton with Karp-Rabin fingerprints validate if matching fingerprints due to matching strings Monte Carlo algorithm

\( O(n(1 + \lg \frac{z}{s})) \) expected time and \( O(n) \) space

only expected construction time!

queries very fast in practice

construction very slow in practice

good topic for thesis

\(~\)space-efficient construction of block trees
Construction of Block Trees

**O(n) Working Space**
- build Aho-Corasick automaton for containing all pairs of consecutive unmarked blocks
- identify unmarked blocks on next level
- $O(n(1 + \lg \frac{z}{s}))$ time and $O(n)$ space

**Pruning**
- size of block tree can be reduced further
- some blocks not necessary
- those blocks can easily be identified
## Construction of Block Trees

### $O(n)$ Working Space
- build Aho-Corasick automaton for containing all pairs of consecutive unmarked blocks
- identify unmarked blocks on next level
- $O(n(1 + \lg \frac{z}{s}))$ time and $O(n)$ space

### $O(s + z\tau)$ Working Space
- replace Aho-Corasick automaton with Karp-Rabin fingerprints
- validate if matching fingerprints due to matching strings
- $O(n(1 + \lg \frac{z}{s}))$ expected time and $O(n)$ space
- only expected construction time!

### Pruning
- size of block tree can be reduced further
- some blocks not necessary
- those blocks can easily be identified
Construction of Block Trees

**O(n) Working Space**
- build Aho-Corasick automaton for containing all pairs of consecutive unmarked blocks
- identify unmarked blocks on next level
- $O(n(1 + \lg \frac{z}{s}))$ time and $O(n)$ space

**Pruning**
- size of block tree can be reduced further
- some blocks not necessary
- those blocks can easily be identified

**O(s + z\tau) Working Space**
- replace Aho-Corasick automaton with Karp-Rabin fingerprints
- validate if matching fingerprints due to matching strings with Monte Carlo algorithm
- $O(n(1 + \lg \frac{z}{s}))$ expected time and $O(n)$ space
- only expected construction time!

- queries very fast in practice
- construction very slow in practice
- good topic for thesis 😊
- space-efficient construction of block trees
Let $T$ be a text, then
- $r(T)$ is number of BWT runs of $T$
- $z(T)$ is number of LZ77 factors of $T$

**Definition: Burrows-Wheeler Transform [BW94]**

Given a text $T$ of length $n$ and its suffix array $SA$, for $i \in [1, n]$ the **Burrows-Wheeler transform** is

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

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>a</td>
<td>b</td>
<td>b</td>
<td>a</td>
<td>$</td>
</tr>
<tr>
<td>$SA$</td>
<td>13</td>
<td>12</td>
<td>1</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>11</td>
<td>2</td>
<td>10</td>
<td>7</td>
<td>4</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>$LCP$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>$BWT$</td>
<td>a</td>
<td>b</td>
<td>$</td>
<td>$</td>
<td>c</td>
<td>c</td>
<td>b</td>
<td>b</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>b</td>
</tr>
</tbody>
</table>
Lemma: Number of BWT Runs

Let $T$ be a text of length $n$, then

$$r(T) \in O(z(T) \lg^2 n)$$

- $LCP[i]$ is irreducible if $i = 1$ or $BWT[i] \neq BWT[i - 1]$
- number of irreducible LCP-values is $r(T)$
Relation Between BWT Runs and LZ Factors (2/3)

Lemma: Number of BWT Runs
Let $T$ be a text of length $n$, then

$$r(T) \in O(z(T) \lg^2 n)$$

- $LCP[i]$ is irreducible if $i = 1$ or $BWT[i] \neq BWT[i - 1]$
- number of irreducible LCP-values is $r(T)$

Lemma: Sum of Irreducible LCP-Values
The number of irreducible LCP-Values in $[\ell, 2\ell)$ is in $O(z\ell \lg n)$
### Lemma: Number of BWT Runs

Let $T$ be a text of length $n$, then

$$r(T) \in O(z(T) \lg^2 n)$$

- $LCP[i]$ is irreducible if $i = 1$ or $BWT[i] \neq BWT[i - 1]$
- number of irreducible LCP-values is $r(T)$

### Proof (Sketch)

- $T^\infty[i] = T[i \mod n]$
- $S_m = \{S \in \Sigma^m : S$ is substring of $T^\infty\}$
- $|S_m| \leq mz$
- for irreducible $LCP[i] \in [\ell, 2\ell)$ charge $\ell$ characters in $S_{3\ell}$
- each string is charged at most $2 \lg n$ time

---

### Lemma: Sum of Irreducible LCP-Values

The number of irreducible LCP-Values in $[\ell, 2\ell)$ is in $O(z\ell \lg n)$
### Lemma: Number of BWT Runs

Let \( T \) be a text of length \( n \), then

\[
 r(T) \in O(z(T) \log^2 n)
\]

- **LCP**[\( i \)] is irreducible if \( i = 1 \) or \( \text{BWT}[i] \neq \text{BWT}[i - 1] \)
- number of irreducible LCP-values is \( r(T) \)

### Lemma: Sum of Irreducible LCP-Values

The number of irreducible LCP-Values in \([\ell, 2\ell)\) is in \( O(\ell \log n) \)

### Proof (Sketch)

- \( T^\infty[i] = T[i \mod n] \)
- \( S_m = \{ S \in \Sigma^m : S \text{ is substring of } T^\infty \} \)
- \( |S_m| \leq m\ell \)
- for irreducible \( \text{LCP}[i] \in [\ell, 2\ell) \) charge \( \ell \) characters in \( S_{3\ell} \)
- each string is charged at most \( 2 \log n \) time

- apply lemma for \([2^i, 2^{i+1})\) for \( i \in [0, \lceil \log n \rceil] \)
- number of \( \text{LCP}[i] = 0 \) entries is \( \sigma \leq z \)
**Lemma: Number of Occurrences of Substrings**

For any $\ell > 1$, the number of distinct substrings of $T$ of length $\ell$ is $\leq z\ell$.

**Proof (Sketch)**

- consider any substring of length $\ell > 1$
- if substring is contained in LZ factor, there is previous occurrence
- distinct substrings overlap LZ factors
- there are at most $\ell$ substring per end of LZ factor

- use number of distinct substrings
- to show that the number of irreducible LCP-values
- is limited as stated in lemma
Conclusion and Outlook

This Lecture
- block trees
- \( r \in O(z \lg^2 n) \)

Linear Time Construction

- ST
- SA
- WT
- LZ
- LCP
- BWT

FM-Index
r-Index
Conclusion and Outlook

This Lecture
- block trees
- \( r \in O(z \log^2 n) \)

Open Questions
- efficient block tree construction
- linear time block tree construction

Linear Time Construction

- ST
- SA
- WT
- LZ
- LCP
- BWT
- FM-Index
- \( r \)-Index
Conclusion and Outlook

This Lecture
- block trees
- $r \in O(z \lg^2 n)$

Open Questions
- efficient block tree construction
- linear time block tree construction

Next Lecture
- suffix array construction in different models of computation

Linear Time Construction

ST \rightarrow SA \rightarrow WT
LZ \rightarrow LCP \rightarrow BWT
FM-Index
r-Index

18/18
2022-12-19 Florian Kurpicz | Text Indexing | 08 LZ- & BWT-Compressed Indices

Institute for Theoretical Informatics, Algorithm Engineering
Bibliography I


