Warning

This is just a very succinct overview.
Please refer to the lecture slides for more details.
Tries & Suffix Trees

Trie Representations
- different trie representations
- space-time trade-off

Suffix Tree (Compact Trie)
Suffix Array

Given a text $T$ of length $n$, the **suffix array** (SA) is a permutation of $[1..n]$, such that for $i \leq j \in [1..n]$

$$T[SA[i]..n] \leq T[SA[j]..n]$$

### SAIS
- linear time suffix array construction
- induced copying and recursion
  - classification
  - sorting special suffixes
  - inducing other suffixes

### SA Construction in EM
- Prefix Doubling
- DC3
### LCP-Array & LCE-Queries

<table>
<thead>
<tr>
<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>
</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>
</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>
</tr>
</tbody>
</table>

- speed up pattern matching in suffix array
- suffix tree construction
- compression

#### Longest Common Extensions
- lcp-value between any suffix
- scan or RMQ
- Rabin-Karp fingerprints
- string synchronizing sets
Compression

Entropy

Given a text $T$ of length $n$ over an alphabet $\Sigma = [1, \sigma]$ and its histogram $Hist$, then

$$H_k = \left(\frac{1}{n}\right) \sum_{S \in \Sigma^k} |T_S| \cdot H_0(T_S)$$

Huffman Codes

- variable length codes
- more frequent characters get shorter codes
- canonical Huffman-codes
- Shannon-Fano codes can be worse, but
- are still optimal

LZ77

$T = ababbbbababa$

- $f_1 = a$
- $f_2 = b$
- $f_3 = abab$
- $f_4 = bbb$
- $f_5 = aba$
- $f_6 = \$

LZ78

$T = ababbbbababa$

- $f_1 = a$
- $f_2 = b$
- $f_3 = ab$
- $f_4 = abb$
- $f_5 = bb$
- $f_6 = aba$
- $f_7 = \$
Burrows-Wheeler Transform

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

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

<table>
<thead>
<tr>
<th>$i$</th>
<th>1 2 3 4 5 6 7 8 9 10 11 12 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>a b a b c a b c a b b a $</td>
</tr>
<tr>
<td>$SA$</td>
<td>13 12 1 9 6 3 11 2 10 7 4 8 5</td>
</tr>
<tr>
<td>$BWT$</td>
<td>a b $ c c b b a a a a b b</td>
</tr>
</tbody>
</table>

LF-Mapping

Given a $BWT$, its $C$-array, and its $rank$-Function, then

$$LF(i) = C[BWT[i]] + rank_{BWT[i]}(i)$$

- transform back to text
- used in backwards search

Compression using BWT

- move-to-front
- run-length compression
Wavelet Tree

- generalize rank and select to alphabets of size > 2

Compression
- build over text compressed with canonical Huffman codes

Bit Vectors
- rank and select queries on bit vectors in $O(1)$ time and $o(n)$ space
Function $\text{BackwardsSearch}(P[1..n], C, \text{rank})$:

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

**FM-Index**

- use (compressed wavelet tree for rank)
- compress bit vectors further

**r-Index**

- store lots of arrays
- containing information for each run
- size proportional to number of runs
- queries become harder
Compressed Indices

Block Tree
- answer rank and select queries
- size proportional to number of LZ-factors

Number of Runs and LZ-Factors
Let $T$ be a text of length $n$, then

$$r(T) \in O(z(T) \log^2 n)$$
Document Retrieval

Document Listing

- optimal with document array and chain array

<table>
<thead>
<tr>
<th>T</th>
<th>A</th>
<th>T</th>
<th>A</th>
<th>#</th>
<th>T</th>
<th>A</th>
<th>A</th>
<th>A</th>
<th>#</th>
<th>T</th>
<th>A</th>
<th>T</th>
<th>A</th>
<th>#</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>6</td>
<td>10</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>SA</td>
<td>15</td>
<td>14</td>
<td>4</td>
<td>9</td>
<td>13</td>
<td>3</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>11</td>
<td>1</td>
<td>12</td>
<td>2</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>DA</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

- $P = TA$

Institute for Theoretical Informatics, Algorithm Engineering
The old night keeper keeps the keep in the town
In the big old house in the big old gown
The house in the town had the big old keep
Where the old night keeper never did sleep
The night keeper keeps the keep in the night
And keeps in the dark and sleeps in the light

<table>
<thead>
<tr>
<th>term</th>
<th>( t_i )</th>
<th>( L(t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>1</td>
<td>[6]</td>
</tr>
<tr>
<td>big</td>
<td>2</td>
<td>[2, 3]</td>
</tr>
<tr>
<td>dark</td>
<td>1</td>
<td>[6]</td>
</tr>
<tr>
<td>had</td>
<td>1</td>
<td>[3]</td>
</tr>
<tr>
<td>house</td>
<td>2</td>
<td>[2, 3]</td>
</tr>
<tr>
<td>in</td>
<td>5</td>
<td>[1, 2, 3, 5, 6]</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Encodings
- unary/ternary encoding
- Fibonacci encoding
- Elias-\( \delta/\gamma \) encoding
- Golomb encoding

List Interseciong
- binary/exponential search
- two levels
Longest Common Extensions

Sophisticated Black Box (BB)
- based on ISA, LCP, and RMQ
- $O(1)$ query time, $\approx 9n$ bytes additional space

Ultra Naive Scan (UNS)
- compare character by character
- $O(n)$ query time, no additional space

Definition: Simplified $\tau$-Synchronizing Sets

Given a text $T$ of length $n$ and $0 < \tau \leq n/2$, a simplified $\tau$-synchronizing set $S$ of $T$ is

$$S = \{ i \in [1, n - 2\tau + 1] : \min\{ \text{block}(j, j + \tau - 1) : j \in [i, i + \tau] \} \in \{ \text{block}(i, i + \tau - 1), \text{block}(i + \tau, i + 2\tau - 1) \} \}$$