

Bzip2

Seminar: Datenkompression

Toprak Saricerci Florian Brohm
7445073 7443251

February 21, 2023

- Verlustfreies Datenkompressionsverfahren

Bzip2: Überblick

- Verlustfreies Datenkompressionsverfahren
- Ähnlich zu ZIP

Bzip2: Überblick

- Verlustfreies Datenkompressionsverfahren
- Ähnlich zu ZIP
- Basiert auf Burrows-Wheeler transform

Bzip2: Überblick

- Verlustfreies Datenkompressionsverfahren
- Ähnlich zu ZIP
- Basiert auf Burrows-Wheeler transform
- Autor: Julian Seward

Bzip2: Überblick

- Verlustfreies Datenkompressionsverfahren
- Ähnlich zu ZIP
- Basiert auf Burrows-Wheeler transform
- Autor: Julian Seward
- Veröffentlichung: 18. Juli 1996

- Verlustfreies Datenkompressionsverfahren
- Ähnlich zu ZIP
- Basiert auf Burrows-Wheeler transform
- Autor: Julian Seward
- Veröffentlichung: 18. Juli 1996
- Verwendete Algorithmen
 - ▶ Run Length Encoding
 - ▶ Huffman Encoding
 - ▶ Move-to-Front Transform
 - ▶ Burrows Wheeler Transform

Was ist eine Transformation?

- Permutationen (Positionsbezogene Abbildung, spez. Rotation)
- Allgemeine Abbildungen
- Länge Output \approx Länge Input
- Invertierbar (verlustfrei)
- **Idee: Gesteigerte Effizienz einer folgenden Enkodierung**

Überblick

- "Runs" von aufeinander folgenden Zeichen werden zusammengefasst
- **Idee: Große Runs sparen viel Platz**

Run Length Kodierung

```
function run_length(input):  
    out  $\leftarrow \epsilon$   
    counter  $\leftarrow 1$   
    run_char  $\leftarrow$  input[1]  
    for i in 2..n:  
        if input[i-1] = input[i]:  
            counter++  
        else:  
            out.append(counter, run_char)  
            counter  $\leftarrow 1$   
            run_char  $\leftarrow$  input[i]  
    out.append(counter)  
    out.append(run_char)  
    return out
```

Überblick

- Relative häufigkeit der Zeichen wird betrachtet um eine Entropie-optimale Kodierungstabelle zu erstellen
- Häufig auftretende Zeichen bekommen kleine Kodierungen
- **Idee: Input wird so kodiert, dass die Länge tatsächlich von dem Informationsgehalt der Quelle abhängig ist**

Huffman Kodierung: Tabelle

```
function huffman_table(input):  
    table: string  $\Rightarrow$  string  
    heap  $\leftarrow$  MinHeap()  
    for char in input:  
        heap.insert(char, occurrence(char))  
    root  $\leftarrow$  {}  
    while heap.hasItem():  
        (char1, char1_occ)  $\leftarrow$  heap.pop()  
        if heap.empty(): break  
        (char2, char2_occ)  $\leftarrow$  heap.pop()  
        root  $\leftarrow$  (char1_occ + char2_occ)  
        heap.insert(root)  
    for leaf in root:  
        table.map(leaf.path  $\rightarrow$  leaf.char)  
    return table
```

```
function huffman_encode(input):  
    table  $\leftarrow$  huffman_table(input)  
    out  $\leftarrow \epsilon$   
    for char in input:  
        out.append(table(char))  
    return table, out
```

```
function huffman_decode(table , input):  
    buffer  $\leftarrow \epsilon$   
    out  $\leftarrow \epsilon$   
    for char in input:  
        buffer.append(char)  
        if buffer in table:  
            out.append(table(buffer))  
            buffer  $\leftarrow \epsilon$   
    return out
```

Überblick

- Transformation des Input
- Input wird sequentiell auf Position in einem Buffer abgebildet
- Zeichendistribution wird auf niedrige Codepoints konzentriert
- **Idee: Codepoints werden öfter wiederverwendet**

Move-to-Front Transform

Sei Σ das Eingabealphabet (typisch: $\Sigma = \text{ASCII}$, $|\Sigma| = 256$).

```
function mtf(input):  
    A[| $\Sigma$ |]  $\leftarrow$   $\Sigma$   
    out  $\leftarrow$   $\epsilon$   
    for char in input:  
        char_index  $\leftarrow$  A.indexOf(char)  
        out.append(char_index)  
        delete A[char_index]  
        prepend(char, A)  
    return out
```


Move-to-Front Transform

```
function inverse_mtf(A, input):  
    out  $\leftarrow \epsilon$   
    for char_index in input:  
        out.append(A[0])  
        swap A[char_index] with A[0]  
    return out
```

Warum ist MTF sinnvoll?

Warum MTF sinnvoll?

MTF hat zwei Vorteile:

Warum MTF sinnvoll?

MTF hat zwei Vorteile:

Vorteil 1:

Wiederholende Zeichenfolgen sind nach MTF gut zu enkodieren.

Warum MTF sinnvoll?

MTF hat zwei Vorteile:

Vorteil 1:

Wiederholende Zeichenfolgen sind nach MTF gut zu enkodieren.

$$(a_1, \dots, a_k)^n \xrightarrow{\text{MTF}} (c_1, \dots, c_k) \cdot (k-1)^{k(n-1)}$$

für $(a_1, \dots, a_k) \in \Sigma^k$ mit $n > 2$ und den Codepoints c_i von a_i .

Warum MTF sinnvoll?

MTF hat zwei Vorteile:

Vorteil 1:

Wiederholende Zeichenfolgen sind nach MTF gut zu enkodieren.

$$(a_1, \dots, a_k)^n \xrightarrow{\text{MTF}} (c_1, \dots, c_k) \cdot (k-1)^{k(n-1)}$$

für $(a_1, \dots, a_k) \in \Sigma^k$ mit $n > 2$ und den Codepoints c_i von a_i .

Input: abcdabcdabcdabcdabcd

Warum MTF sinnvoll?

MTF hat zwei Vorteile:

Vorteil 1:

Wiederholende Zeichenfolgen sind nach MTF gut zu enkodieren.

$$(a_1, \dots, a_k)^n \xrightarrow{\text{MTF}} (c_1, \dots, c_k) \cdot (k-1)^{k(n-1)}$$

für $(a_1, \dots, a_k) \in \Sigma^k$ mit $n > 2$ und den Codepoints c_i von a_i .

Input: abcdabcdabcdabcdabcdabcd

Output: 65 66 67 68 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

Warum MTF sinnvoll?

MTF hat zwei Vorteile:

Vorteil 2:

Warum MTF sinnvoll?

MTF hat zwei Vorteile:

Vorteil 2:

- Häufige Zeichen sind links im Alphabet.
- Transformation erzeugt viele kleine Zahlen.
- Großteil des Outputs W besteht aus Zeichen $Z = \{1, \dots, 9\}$.
- $|W| \gg |Z| \Rightarrow$ Huffman und Run-Length sind effektiver.

Überblick

- Transformation des Input
- Input wird basierend auf lexikographischen Vergleichen permutiert
- **Idee: Durchschnittliche Run-Length wird vergrößert**

Burrows-Wheeler Transform

```
function burrows_wheeler(input):  
    rotations[n, n]  $\leftarrow$  input.rotations()  
    rotations.sort()  
    return rotations[1..n, n]
```

Burrows-Wheeler Transform

```
function inverse_burrows_wheeler(input):  
    rotations[n, n]  
    for k in n..1:  
        rotations[1..n, k]  $\leftarrow$  input  
        rotations.sort()  
    return rotation ending with $
```

Warum ist BWT sinnvoll?

Warum ist BWT sinnvoll?

Folgender Satz demonstriert die Stärke des BWT:

eulen heulen wegen beulen

Warum ist BWT sinnvoll?

Folgender Satz demonstriert die Stärke des BWT:

eulen heulen wegen beulen

Hier fällt auf:

Warum ist BWT sinnvoll?

Folgender Satz demonstriert die Stärke des BWT:

eul**e**n heul**e**n weg**e**n beul**e**n

Hier fällt auf:

- Vor "n" ist immer ein "e".

Warum ist BWT sinnvoll?

Folgender Satz demonstriert die Stärke des BWT:

eulen heulen wegen beulen

Hier fällt auf:

- Vor "n" ist immer ein "e".
- Vor " " ist immer ein "n".

Warum ist BWT sinnvoll?

Folgender Satz demonstriert die Stärke des BWT:

eulen heulen wegen beulen

Hier fällt auf:

- Vor "n" ist immer ein "e".
- Vor " " ist immer ein "n".
- Vor "e" ist immer ein "u".

Warum ist BWT sinnvoll?

Folgender Satz demonstriert die Stärke des BWT:

eul^{en} heul^{en} wegen beul^{en}

Hier fällt auf:

- Vor "n" ist immer ein "e".
- Vor " " ist immer ein "n".
- Vor "e" ist immer ein "u".
- Vor "u" ist immer ein "l".

Warum ist BWT sinnvoll?

Folgender Satz demonstriert die Stärke des BWT:

eul^{en} heul^{en} wegen beul^{en}

Hier fällt auf:

- Vor "n" ist immer ein "e".
- Vor " " ist immer ein "n".
- Vor "e" ist immer ein "u".
- Vor "u" ist immer ein "l".

Diese Regelmäßigkeiten wollen wir nutzen!

Warum ist BWT sinnvoll?

eulen heulen wegen beulen

Warum ist BWT sinnvoll?

eulen heulen wegen beulen

eulen heulen wegen beulen\$
\$eulen heulen wegen beulen
n\$eulen heulen wegen beule
en\$eulen heulen wegen beul
len\$eulen heulen wegen beu
ulen\$eulen heulen wegen be
eulen\$eulen heulen wegen b
beulen\$eulen heulen wegen
beulen\$eulen heulen wegen
n beulen\$eulen heulen wege
en beulen\$eulen heulen weg
gen beulen\$eulen heulen we
egen beulen\$eulen heulen w
wegen beulen\$eulen heulen
wegen beulen\$eulen heulen
n wegen beulen\$eulen heule
en wegen beulen\$eulen heul
len wegen beulen\$eulen heu
ulen wegen beulen\$eulen he
eulen wegen beulen\$eulen h
heulen wegen beulen\$eulen
heulen wegen beulen\$eulen
n heulen wegen beulen\$eule
en heulen wegen beulen\$eul
len heulen wegen beulen\$eu
ulen heulen wegen beulen\$e

Warum ist BWT sinnvoll?

eulen heulen wegen beulen

eulen heulen wegen beulen\$
\$eulen heulen wegen beulen
n\$eulen heulen wegen beule
en\$eulen heulen wegen beul
len\$eulen heulen wegen beu
ulen\$eulen heulen wegen be
eulen\$eulen heulen wegen b
beulen\$eulen heulen wegen
beulen\$eulen heulen wegen
n beulen\$eulen heulen wege
en beulen\$eulen heulen weg
gen beulen\$eulen heulen we
egen beulen\$eulen heulen w
wegen beulen\$eulen heulen
wegen beulen\$eulen heulen
n wegen beulen\$eulen heule
en wegen beulen\$eulen heul
len wegen beulen\$eulen heu
ulen wegen beulen\$eulen he
eulen wegen beulen\$eulen h
heulen wegen beulen\$eulen
heulen wegen beulen\$eulen
n heulen wegen beulen\$eule
en heulen wegen beulen\$eul
len heulen wegen beulen\$eu
ulen heulen wegen beulen\$e

beulen\$eulen heulen wegen
heulen wegen beulen\$eulen
wegen beulen\$eulen heulen
beulen\$eulen heulen wegen_
egen beulen\$eulen heulen w
en beulen\$eulen heulen weg
en heulen wegen beulen\$eul
en wegen beulen\$eulen heul
en\$eulen heulen wegen beul
eulen heulen wegen beulen\$
eulen wegen beulen\$eulen h
eulen\$eulen heulen wegen b
gen beulen\$eulen heulen we
heulen wegen beulen\$eulen_
len heulen wegen beulen\$eu
len wegen beulen\$eulen heu
len\$eulen heulen wegen beu
n beulen\$eulen heulen wege
n heulen wegen beulen\$eule
n wegen beulen\$eulen heule
n\$eulen heulen wegen beule
ulen heulen wegen beulen\$e
ulen wegen beulen\$eulen he
ulen\$eulen heulen wegen be
wegen beulen\$eulen heulen_
\$eulen heulen wegen beulen

Warum ist BWT sinnvoll?

beulen\$eulen heulen wegen_
heulen wegen beulen\$eulen_
wegen beulen\$eulen heulen_
beulen\$eulen heulen wegen_
egen beulen\$eulen heulen w_
en beulen\$eulen heulen weg
en heulen wegen beulen\$eu_
en wegen beulen\$eulen heu_
en\$eulen heulen wegen beu_
eulen heulen wegen beulen\$
eulen wegen beulen\$eulen h_
eulen\$eulen heulen wegen b_
gen beulen\$eulen heulen we_
heulen wegen beulen\$eulen_
len heulen wegen beulen\$eu
len wegen beulen\$eulen heu_
len\$eulen heulen wegen beu_
n beulen\$eulen heulen wege_
n heulen wegen beulen\$eule_
n wegen beulen\$eulen heule_
n\$eulen heulen wegen beul_
ulen heulen wegen beulen\$e_
ulen wegen beulen\$eulen he_
ulen\$eulen heulen wegen be_
wegen beulen\$eulen heulen_
\$eulen heulen wegen beulen_

Warum ist BWT sinnvoll?

beulen§eulen heulen wegen_
heulen wegen beulen§eulen_
wegen beulen§eulen heulen_
beulen§eulen heulen wegen_
egen beulen§eulen heulen w
en beulen§eulen heulen w
en heulen wegen beulen§eul
en wegen beulen§eulen heul
en§eulen heulen wegen beul
eulen heulen wegen beulen§
eulen wegen beulen§eulen h
eulen§eulen heulen wegen b
gen beulen§eulen heulen w
heulen wegen beulen§eulen_
len heulen wegen beulen§eu
len wegen beulen§eulen heu
len§eulen heulen wegen beu
n beulen§eulen heulen weg
n heulen wegen beulen§eul
n wegen beulen§eulen heul
n§eulen heulen wegen beul
ulen heulen wegen beulen§
ulen wegen beulen§eulen h
ulen§eulen heulen wegen b
wegen beulen§eulen heulen_
§eulen heulen wegen beulen

Input: eulen heulen wegen beulen
Output: nnn wglll§hbe uuueeeeeeee n

Warum ist BWT sinnvoll?

beulen\$eulen heulen wegen
heulen wegen beulen\$eulen
wegen beulen\$eulen heulen
beulen\$eulen heulen wegen
egen beulen\$eulen heulen w
en beulen\$eulen heulen weg
en heulen wegen beulen\$eu
en wegen beulen\$eulen heul
en\$eulen heulen wegen beul
eulen heulen wegen beulen\$
eulen wegen beulen\$eulen h
eulen\$eulen heulen wegen b
gen beulen\$eulen heulen we
heulen wegen beulen\$eulen
len heulen wegen beulen\$eu
len wegen beulen\$eulen heu
len\$eulen heulen wegen beu
n beulen\$eulen heulen wege
n heulen wegen beulen\$eule
n wegen beulen\$eulen heule
n\$eulen heulen wegen beule
ulen heulen wegen beulen\$e
ulen wegen beulen\$eulen he
ulen\$eulen heulen wegen be
wegen beulen\$eulen heulen
\$eulen heulen wegen beulen

Input: eulen heulen wegen beulen
Output: nnn wgl||\$hbe uuueeeeeee n

Betrachte die roten Zeilen:

Warum ist BWT sinnvoll?

beulen\$eulen heulen wegen_
heulen wegen beulen\$eulen_
wegen beulen\$eulen heulen_
beulen\$eulen heulen wegen_
egen beulen\$eulen heulen w_
en beulen\$eulen heulen weg
en heulen wegen beulen\$eu_
en wegen beulen\$eulen heu_
en\$eulen heulen wegen beu_
eulen heulen wegen beulen\$
eulen wegen beulen\$eulen h_
eulen\$eulen heulen wegen b_
gen beulen\$eulen heulen we_
heulen wegen beulen\$eulen_
len heulen wegen beulen\$eu_
len wegen beulen\$eulen heu_
len\$eulen heulen wegen beu_
n beulen\$eulen heulen wege
n heulen wegen beulen\$eule
n wegen beulen\$eulen heule
n\$eulen heulen wegen beule
ulen heulen wegen beulen\$e
ulen wegen beulen\$eulen he
ulen\$eulen heulen wegen be
wegen beulen\$eulen heulen_
\$eulen heulen wegen beulen

Input: eulen heulen wegen beulen
Output: nnn wgl||\$hbe uuueeeeeee n

Betrachte die roten Zeilen:

- n ist der erste Buchstabe der Zeilen

Warum ist BWT sinnvoll?

beulenßeulen heulen wegen
heulen wegen beulenßeulen
wegen beulenßeulen heulen
beulenßeulen heulen wegen
egen beulenßeulen heulen w
en beulenßeulen heulen weg
en heulen wegen beulenßeul
en wegen beulenßeulen heul
enßeulen heulen wegen beul
eulen heulen wegen beulenße
eulen wegen beulenßeulen h
eulenßeulen heulen wegen b
gen beulenßeulen heulen we
heulen wegen beulenßeulen
len heulen wegen beulenßeu
len wegen beulenßeulen heu
lenßeulen heulen wegen beu
n beulenßeulen heulen wege
n heulen wegen beulenßeule
n wegen beulenßeulen heule
nßeulen heulen wegen beule
ulen heulen wegen beulenße
ulen wegen beulenßeulen he
ulenßeulen heulen wegen be
wegen beulenßeulen heulen
ßeulen heulen wegen beulen

Input: eulen heulen wegen beulen
Output: nnn wgl||ßhbe uuueeeeeee n

Betrachte die roten Zeilen:

- n ist der erste Buchstabe der Zeilen
 - Im Satz ist vor n immer ein e

Warum ist BWT sinnvoll?

beulenßeulen heulen wegen
heulen wegen beulenßeulen
wegen beulenßeulen heulen
beulenßeulen heulen wegen
egen beulenßeulen heulen w
en beulenßeulen heulen weg
en heulen wegen beulenßeul
en wegen beulenßeulen heul
enßeulen heulen wegen beul
eulen heulen wegen beulenße
eulen wegen beulenßeulen h
eulenßeulen heulen wegen b
gen beulenßeulen heulen we
heulen wegen beulenßeulen
len heulen wegen beulenßeu
len wegen beulenßeulen heu
lenßeulen heulen wegen beu
n beulenßeulen heulen wege
n heulen wegen beulenßeule
n wegen beulenßeulen heule
nßeulen heulen wegen beule
ulen heulen wegen beulenße
ulen wegen beulenßeulen he
ulenßeulen heulen wegen be
wegen beulenßeulen heulen
ßeulen heulen wegen beulen

Input: eulen heulen wegen beulen
Output: nnn wgl||ßhbe uuueeeeeee n

Betrachte die roten Zeilen:

- n ist der erste Buchstabe der Zeilen
 - ▶ Im Satz ist vor n immer ein e
 - ▶ Alle e sind in aufeinanderfolgenden Zeilen

Warum ist BWT sinnvoll?

beulenßeulen heulen wegen
heulen wegen beulenßeulen
wegen beulenßeulen heulen
beulenßeulen heulen wegen
egen beulenßeulen heulen w
en beulenßeulen heulen weg
en heulen wegen beulenßeul
en wegen beulenßeulen heul
enßeulen heulen wegen beul
eulen heulen wegen beulenße
eulen wegen beulenßeulen h
eulenßeulen heulen wegen b
gen beulenßeulen heulen we
heulen wegen beulenßeulen
len heulen wegen beulenßeu
len wegen beulenßeulen heu
lenßeulen heulen wegen beu
n beulenßeulen heulen wege
n heulen wegen beulenßeule
n wegen beulenßeulen heule
nßeulen heulen wegen beule
ulen heulen wegen beulenße
ulen wegen beulenßeulen he
ulenßeulen heulen wegen be
wegen beulenßeulen heulen
ßeulen heulen wegen beulen

Input: eulen heulen wegen beulen
Output: nnn wgl||\$hbe uuueeeeeee n

Betrachte die roten Zeilen:

- n ist der erste Buchstabe der Zeilen
 - ▶ Im Satz ist vor n immer ein e
 - ▶ Alle e sind in aufeinanderfolgenden Zeilen
- Ähnlich für

Warum ist BWT sinnvoll?

beulenßeulen heulen wegen
heulen wegen beulenßeulen
wegen beulenßeulen heulen
beulenßeulen heulen wegen
egen beulenßeulen heulen w
en beulenßeulen heulen weg
en heulen wegen beulenßeul
en wegen beulenßeulen heul
enßeulen heulen wegen beul
eulen heulen wegen beulenß
eulen wegen beulenßeulen h
eulenßeulen heulen wegen b
gen beulenßeulen heulen we
heulen wegen beulenßeulen
len heulen wegen beulenßeu
len wegen beulenßeulen heu
lenßeulen heulen wegen beu
n beulenßeulen heulen wege
n heulen wegen beulenßeul
n wegen beulenßeulen heule
nßeulen heulen wegen beul
ulen heulen wegen beulenße
ulen wegen beulenßeulen he
ulenßeulen heulen wegen be
wegen beulenßeulen heulen
ßeulen heulen wegen beulen

Input: eulen heulen wegen beulen
Output: nnn wglllßhbe uuueeeeeeee n

Betrachte die roten Zeilen:

- **n** ist der erste Buchstabe der Zeilen
 - ▶ Im Satz ist vor **n** immer ein **e**
 - ▶ Alle **e** sind in aufeinanderfolgenden Zeilen
- Ähnlich für **u**

Warum ist BWT sinnvoll?

```
beulen$eulen heulen wegen_
heulen wegen beulen$eulen_
wegen beulen$eulen heulen_
beulen$eulen heulen wegen_
egen beulen$eulen heulen _
en beulen$eulen heulen weg
en heulen wegen beulen$eu_
en wegen beulen$eulen heul_
en$eulen heulen wegen beu_
eulen heulen wegen beulen$
eulen wegen beulen$eulen h_
eulen$eulen heulen wegen b_
gen beulen$eulen heulen we_
heulen wegen beulen$eulen_
len heulen wegen beulen$eu_
len wegen beulen$eulen heu_
len$eulen heulen wegen beu_
n beulen$eulen heulen wege_
n heulen wegen beulen$eule_
n wegen beulen$eulen heule_
n$eulen heulen wegen beule_
ulen heulen wegen beulen$e_
ulen wegen beulen$eulen he_
ulen$eulen heulen wegen be_
wegen beulen$eulen heulen_
$eulen heulen wegen beulen_
```

Input: eulen heulen wegen beulen
Output: nnn wglll\$hbbe uuueeeeeeee n

Betrachte die roten Zeilen:

- **n** ist der erste Buchstabe der Zeilen
 - ▶ Im Satz ist vor **n** immer ein **e**
 - ▶ Alle **e** sind in aufeinanderfolgenden Zeilen
- Ähnlich für **u**, **e**

Warum ist BWT sinnvoll?

beulen\$eulen heulen wegen
heulen wegen beulen\$eulen
wegen beulen\$eulen heulen

beulen\$eulen heulen wegen_
egen beulen\$eulen heulen w
en beulen\$eulen heulen weg
en heulen wegen beulen\$eu
en wegen beulen\$eulen heul
en\$eulen heulen wegen beul
eulen heulen wegen beulen\$
eulen wegen beulen\$eulen h
eulen\$eulen heulen wegen b
gen beulen\$eulen heulen we
heulen wegen beulen\$eulen_
len heulen wegen beulen\$eu
len wegen beulen\$eulen heu
len\$eulen heulen wegen beu
n beulen\$eulen heulen wege
n heulen wegen beulen\$eul
n wegen beulen\$eulen heule
n\$eulen heulen wegen beul
ulen heulen wegen beulen\$e
ulen wegen beulen\$eulen h
ulen\$eulen heulen wegen be
wegen beulen\$eulen heulen_
\$eulen heulen wegen beulen

Input: eulen heulen wegen beulen
Output: nnn wglll\$hb e uuuuuuuuuuu n

Betrachte die roten Zeilen:

- **n** ist der erste Buchstabe der Zeilen
 - ▶ Im Satz ist vor **n** immer ein **e**
 - ▶ Alle **e** sind in aufeinanderfolgenden Zeilen
- Ähnlich für **u**, **e** und **n**.

Warum ist der BWT sinnvoll?

Input: eulen heulen wegen beulen
Output: nnn wgl||§hbe uuueeeeeeee n

Warum ist der BWT sinnvoll?

Input: eulen heulen wegen beulen
Output: nnn wgl||§hbe uuueeeeeeee n

Die Run-Length Encodierung ergibt folgendes:

Warum ist der BWT sinnvoll?

Input: eulen heulen wegen beulen
Output: nnn wglll§hbe uuueeeeeeee n

Die Run-Length Enkodierung ergibt folgendes:

Input \xRightarrow{RL} 1e1u1l1e1n1 1h1e1u1l1e1n1 1w1e1g1e1n1 1b1e1u1l1e1n

Warum ist der BWT sinnvoll?

Input: eulen heulen wegen beulen
Output: nnn wglll§hbe uuueeeeeeee n

Die Run-Length Encodierung ergibt folgendes:

Input \xRightarrow{RL} 1e1u1l1e1n1 1h1e1u1l1e1n1 1w1e1g1e1n1 1b1e1u1l1e1n
Output \xRightarrow{RL} 3n1 1w1g3l1§1h1b1e1 3u7e1 1n

Warum ist der BWT sinnvoll?

Input: eulen heulen wegen beulen
Output: nnn wglll§hbe uuueeeeeeee n

Die Run-Length Encodierung ergibt folgendes:

Input \xRightarrow{RL} 1e1u1l1e1n1 1h1e1u1l1e1n1 1w1e1g1e1n1 1b1e1u1l1e1n
Output \xRightarrow{RL} 3n1 1w1g3l1§1h1b1e1 3u7e1 1n

Wann ist BWT also sinnvoll:

Warum ist der BWT sinnvoll?

Input: eulen heulen wegen beulen
Output: nnn wglll§hbe uuueeeeeeee n

Die Run-Length Encodierung ergibt folgendes:

Input \xRightarrow{RL} 1e1u1l1e1n1 1h1e1u1l1e1n1 1w1e1g1e1n1 1b1e1u1l1e1n
Output \xRightarrow{RL} 3n1 1w1g3l1§1h1b1e1 3u7e1 1n

Wann ist BWT also sinnvoll:

- Bei Texten mit vielen gleichen Buchstabenfolgen.

Warum ist der BWT sinnvoll?

Input: eulen heulen wegen beulen
Output: nnn wglll§hbe uuueeeeeeee n

Die Run-Length Encodierung ergibt folgendes:

Input \xRightarrow{RL} 1e1u1l1e1n1 1h1e1u1l1e1n1 1w1e1g1e1n1 1b1e1u1l1e1n
Output \xRightarrow{RL} 3n1 1w1g3l1§1h1b1e1 3u7e1 1n

Wann ist BWT also sinnvoll:

- Bei Texten mit vielen gleichen Buchstabenfolgen.
- Gibt es in Sprachen sehr oft!

Vielen Dank fürs Zuhören!