



DATEN KOMPRIMIEREN

M114 / ARJ / 5-2025

Gibt es noch Fragen zu DATEN CODIEREN?

- Massvorsätze k,M,G,T ki,Mi,Gi,Ti
- Bit und Byte
- AND / OR / NOT / XOR
- Parallele/Serielle Datenübertragung
- Taktsignal bei der Datenübertragung
- Zahlensysteme BIN / DEZ / HEX
- Binär addieren, DataOverflow
- Negative Binärzahlen mit 2-er Komplement
- Fliesskommazahlenformat
- Online-HEX-Editor hexed.it, Notepad++
- Alphanumerische Codes ASCII, ANSI-ASCII, ISO8859-x
- Unicode, Unicode V2, UTF8, UTF16 LE-BOM und RE-BOM (Byte-Order)

Lernziele DATEN KOMPRIMIEREN

Verlustlose Komprimierungsverfahren verstehen und anwenden:

- Variable-Length-Coding VLC (Huffman)
- Run-Length-Encoding RLE
- Lempel-Ziv-Welch LZW bei z.B. GIF und TIF
- BWT (Burrows-Wheeler-Transformation)
- ZIP (Zipper bedeutet Reissverschluss)
(Siehe auch Deflate, LZ77, Huffman)

*(Verlustbehaftete Verfahren behandeln wir später
beim Thema "Bilder codieren und komprimieren")*

Verlustlos Komprimieren

Ziel: Volumen verkleinern



...sollen in den Karton passen



Komprimiert
Benötigt nun weniger Platz



Dekomprimiert
Entspricht wieder 1:1 dem Original

Beispiele



ZIP



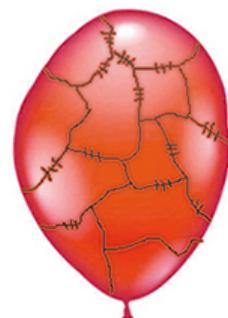
PNG

Verlustbehaftet Komprimieren

Ziel: Volumen verkleinern



Komprimiert
Benötigt nun weniger Platz



Dekomprimiert
Entspricht nicht mehr 1:1 dem Original

Man hat somit einen Verlust erfahren

Beispiele



MP3

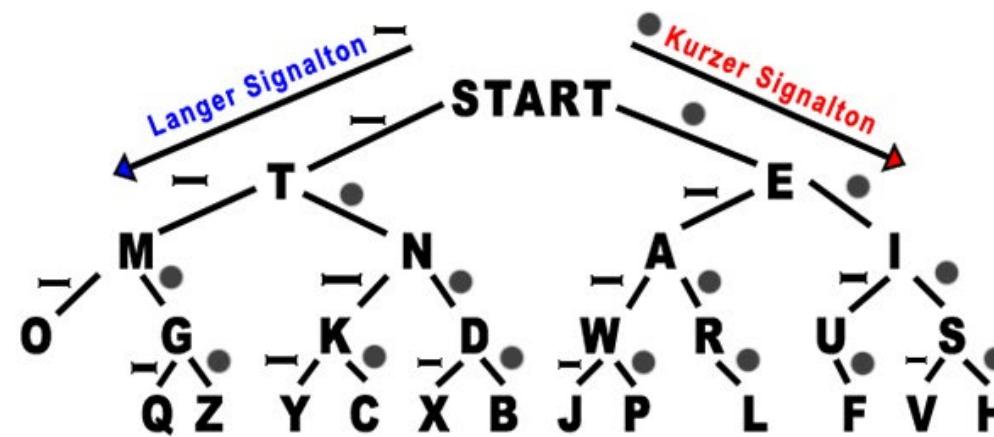


JPG

Code zur telegrafischen Übermittlung von Text: Der Morsecode



| | | | | | |
|---|---------|---|---------|---|---------|
| A | • - | J | • - - - | S | • • • |
| B | - • • • | K | - • - | T | - |
| C | - • - • | L | • - • • | U | • • - |
| D | - • • | M | - - | V | • • • - |
| E | • | N | - • | W | • - - |
| F | • • - • | O | - - - | X | - • • - |
| G | - - • | P | • - - • | Y | - • - - |
| H | • • • • | Q | - - • - | Z | - - • • |
| I | • • | R | • - • | | |



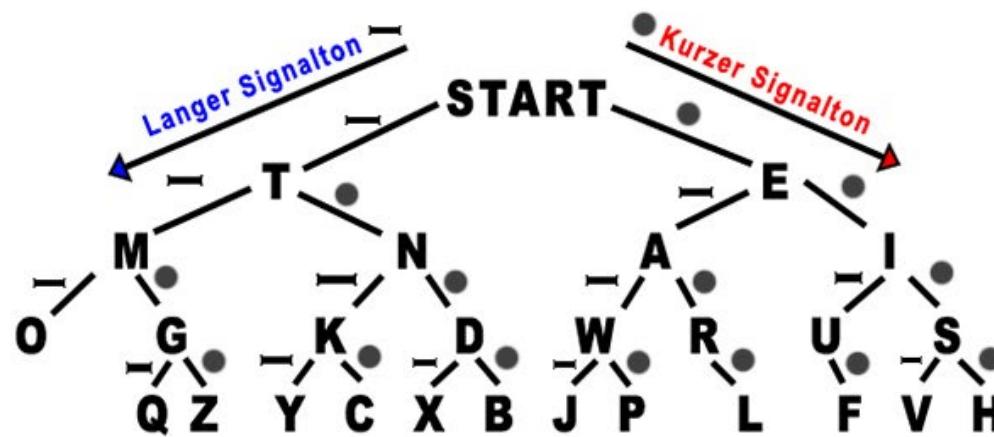
Was sagt uns dieser Morsecode? <https://www.juergarnold.ch/Kompression/Morsecode.wav>



Code zur telegrafischen Übermittlung von Text: Der Morsecode



| | | | | | |
|---|---------|---|---------|---|---------|
| A | • - | J | • --- | S | • • • |
| B | - • • • | K | - • - | T | - |
| C | - • - • | L | • - • • | U | • • - |
| D | - • • | M | - - | V | • • • - |
| E | • | N | - • | W | • - - |
| F | • • - • | O | - - - | X | - • - • |
| G | - - • | P | • - - • | Y | - • - - |
| H | • • • • | Q | - - - • | Z | - - • • |
| I | • • | R | • - - | | |



Der Morsecode hat eine variable Codelänge. VLC oder Variable Length Code.

Das in deutschen und englischen Texten am häufigsten auftretende E entspricht der Codelänge 1.

Das eher seltene Y entspricht der Codelänge 4.



Der Morsecode weist allerdings eine Schwachstelle auf, wie das folgende Beispiel zeigt:

• • - -

Dies kann nämlich folgendes bedeuten:

IDEA ... -... . .-

USA ...- -

UHT ...- - (UHT=UltraHighTemperatur)

FV ...-. - (FV=FussballVerein)

| | | | | | |
|---|-------|---|-------|---|------|
| A | •- | J | •--- | S | ••• |
| B | -••• | K | -•- | T | - |
| C | -••- | L | •-•• | U | ••- |
| D | -•• | M | -- | V | •••- |
| E | • | N | -• | W | •-- |
| F | ••-• | O | --- | X | -••- |
| G | ----• | P | •---• | Y | -•-- |
| H | •••• | Q | -•-•- | Z | -••• |
| I | •• | R | •-• | | |

Beim Morsecode wird die Eindeutigkeit durch ein "Trennzeichen" gelöst. In diesem Fall durch eine kurze Pause nach jedem Buchstaben.

Wir werden dies mit dem Hufmann-Code "verbessern" und damit ohne Trennzeichen auskommen.

Das Konzept des Morsecode (binärer Baum) in einer verbesserten Variante:

Huffman-Code (VLC=Variable Codelänge)

Der Huffman-Code ist im Gegensatz zum Morsecode auch ohne Trennzeichen eindeutig!

Link: https://www.w3schools.com/dsa/dsa_ref_huffman_coding.php

1. Wir werden nun den Huffman-Code für explizit das folgende Wort erstellen:



ERDBEERE

Beispiel mit **8 Zeichen**, wobei

- **E viermal** vorhanden ist
- **R zweimal** vorhanden ist
- **D und B je einmal** vorhanden sind



ERDBEERE

1
Wir werden jetzt den binären Baum rückwärts aufbauen...

| | | | |
|---|---|---|---|
| E | R | D | B |
| 4 | 2 | 1 | 1 |

2

| | | | |
|---|---|---|---|
| E | R | D | B |
| 4 | 2 | X | X |

Tipp: Bereits verwendete Elemente streiche ich durch

3

| | | | |
|---|---|---|---|
| E | R | D | B |
| 4 | X | X | X |

4

Kontrolle:
ERDBEERE besteht aus 8 Buchstaben

| | | | |
|---|---|---|---|
| E | R | D | B |
| X | X | X | X |

5

Zweige mit 0 und 1 beschriften:
Auf welchem Ast 0 oder 1 steht, spielt bei günstiger Codeeffizienz keine Rolle!

| | | | |
|---|---|---|---|
| E | R | D | B |
| 4 | 2 | 1 | 1 |

6

| | | | |
|---|---|---|---|
| E | R | D | B |
| 4 | 2 | 1 | 1 |

Codetabelle:

E = 1
R = 01
D = 001
B = 000

Codebuch (Codetabelle) erstellen.

| | | | | | | | |
|---|----|-----|-----|---|---|----|---|
| E | R | D | B | E | E | R | E |
| 1 | 01 | 001 | 000 | 1 | 1 | 01 | 1 |

(14 Bit)

2. Der binäre Baum für "ERDBEERE" ist eindeutig. Erweitert man die ERDBEERE um ein "N", sieht das schon ganz anders aus:



Beispiel mit **9 Zeichen**, wobei

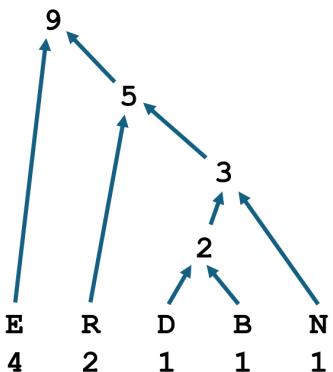
- **E viermal** vorhanden ist
- **R zweimal** vorhanden ist
- **D,B,N** je **einmal** vorhanden sind



ERDBEEREN

Es sind fünf gleichwertige binäre Bäume möglich. Man muss sich für eine Variante entscheiden.

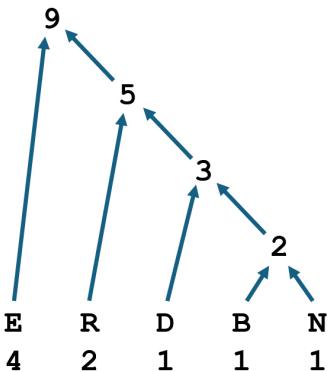
Die Codeeffizienz ist bei allen fünf Varianten dieselbe:
19 Bit.



1. Variante

E = 1
R = 01
D = 0011
B = 0010
N = 000

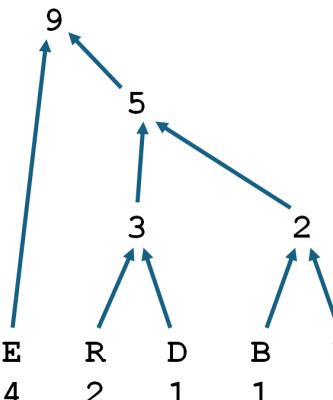
E R D B E E R E N
1 01 0011 0010 1 1 01 1 000
(19 Bit)



2. Variante

E = 1
R = 01
D = 001
B = 0001
N = 0000

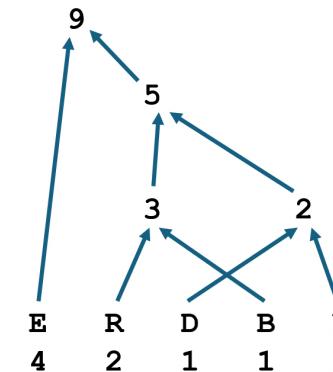
E R D B E E R E N
1 01 001 0001 1 1 01 1 0000
(19 Bit)



3. Variante

E = 1
R = 011
D = 010
B = 001
N = 000

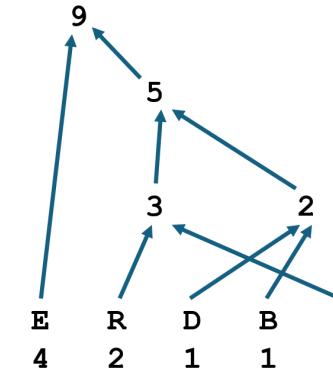
E R D B E E R E N
1 011 010 001 1 1 011 1 000
(19 Bit)



4. Variante

E = 1
R = 011
D = 001
B = 010
N = 000

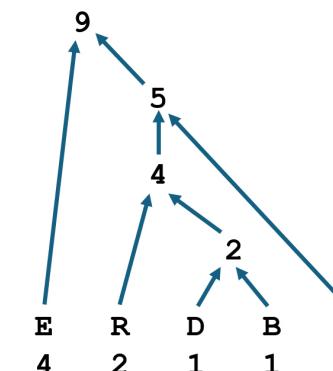
E R D B E E R E N
1 011 001 010 1 1 011 1 000
(19 Bit)



5. Variante

E = 1
R = 011
D = 001
B = 000
N = 010

E R D B E E R E N
1 011 001 000 1 1 011 1 010
(19 Bit)



Falsche Variante

E = 1
R = 011
D = 0101
B = 0100
N = 00

E R D B E E R E N
1 011 0101 0100 1 1 011 1 00
(20 Bit)

Huffman-Aufgabe Zeit: 12 Min.

Erstellen sie den Huffman-Baum und den Huffman-Code für das 12 Buchstaben lange Wort:
(Bitte diesmal keine Huffman-APP benutzen.)

BERUFGSSCHULE

Zeichnen sie den Baum von Hand.

Die Handzeichnung mit einem Smartphone einscannen und
als JPG oder PNG auf Teams (Aufgabe) abgeben.

(Zeichnung per Zeichnungsapplikation nur, wenn sie das Programm auch wirklich beherrschen und die Zeichnung innerhalb des Zeitlimits erstellen und auch als JPG hochladen können.)



MUSTERLÖSUNG

Eine mögliche Lösung. Andere Lösungen müssen ebenfalls 38 Bit ergeben!

Codetabelle:

E=11

U=10

S=011

B=0101

R=0100

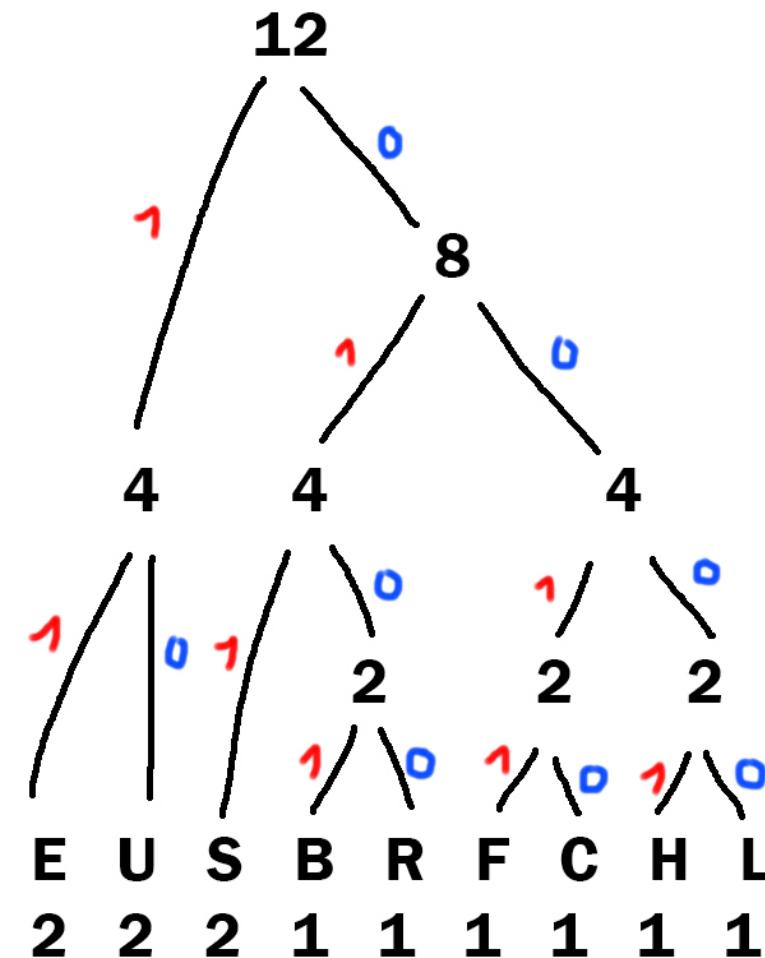
F=0011

C=0010

H=0001

L=0000

BERUFSSCHULE = 12 CHAR

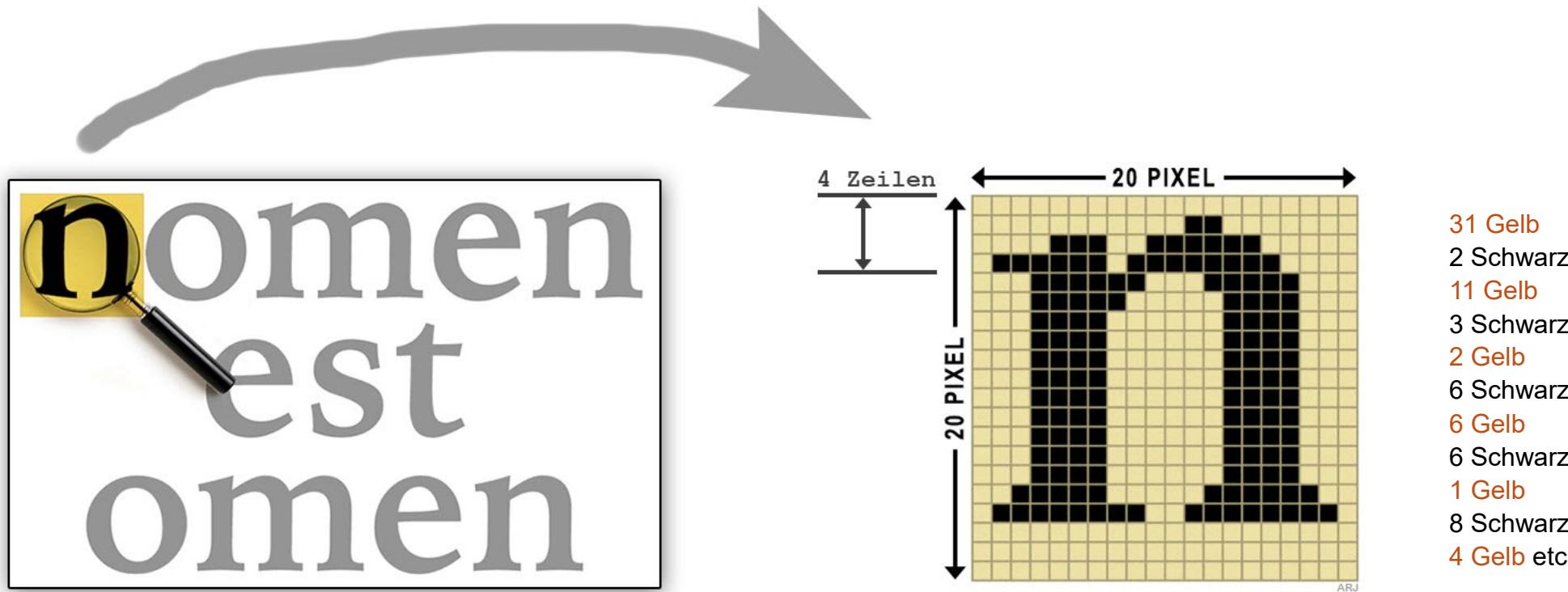


B E R U F S S C H U L E
01011101001000110110110010000110000011

Summe: 38 Bit



RLC (RUN LENGTH CODING) RLE (RUN LENGTH ENCODING)



Betrachtet man die ersten 4 Zeilen: Anzahl_{Max.} = 31, Somit werden 5 Bit pro Anzahl benötigt.

DEZ: 31 2 11 3 2 6 6 6 1 8 4 etc.

BIN: 11111 00010 01011 00011 00010 00110 00110 00110 00001 01000 00100 etc.

Link: <https://planetcalc.com/8670>

RLE-Aufgabe

Zeit: 5 Min.

Sie erhalten diesen RL-Code:

Folgendes ist ihnen dazu bekannt:

- Es handelt sich um eine quadratische Schwarz-Weiss-Rastergrafik mit einer Kantenlnge von 8 Pixeln
 - Das erste Pixel liegt links oben und hat die Farbe Weiss
 - Ein Pixel mit selber Farbe kann sich nicht mehr als siebenmal wiederholen

Zeichnen sie diese Grafik. Was stellt sie dar?



MUSTERLÖSUNG:

Sie erhalten diesen RL-Code:

Dezimal: 2 4 3 6 2 2 2 2 2 2 2 2 2 6 2 6 2 2 2 2 2 2 2 2 2 1

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |
| 6 | | | | | | | | |
| 7 | | | | | | | | |
| 8 | | | | | | | | |

Quadratische Schwarz-Weiss-Rastergrafik

Kantenlnge von 8 Pixel

8x8 Pixel Quadrat

Start links oben mit der Farbe Weiss

Eine Farbe kann sich nicht mehr als siebenmal wiederholen bedeutet:

$0..7 = 3 \text{ Bit oder } 000 \text{ bis } 111$

Dargestellt wird der Grossbuchstabe A



Der LZW-Code Lempel-Ziv-Welch

Idee: Referenzieren, anstatt doppelt abspeichern.
Aufbau eines temporären Wörterbuchs.

LZW-Komprimierung und Dekomprimierung erfolgen auf ähnliche Weise.

LZW-Code

Komprimierung

| Schritt | Zeichenkette | Gefunden | Gespeichert | Temporärer Wörterbucheintrag |
|---------|------------------|-------------|-------------|------------------------------|
| 1. | ENTGEGENGENOMMEN | E | E | EN → «256» |
| 2. | ENTGEGENGENOMMEN | N | N | NT → «257» |
| 3. | ENTGEGENGENOMMEN | T | T | TG → «258» |
| 4. | ENTGEGENGENOMMEN | G | G | GE → «259» |
| 5. | ENTGEGENGENOMMEN | E | E | EG → «260» |
| 6. | ENTGEGENGENOMMEN | GE → «259» | «259» | GEN → «261» |
| 7. | ENTGEGENGENOMMEN | N | N | NG → «262» |
| 8. | ENTGEGENGENOMMEN | GEN → «261» | «261» | GENO → «263» |
| 9. | ENTGEGENGENOMMEN | O | O | OM → «264» |
| 10. | ENTGEGENGENOMMEN | M | M | MM → «265» |
| 11. | ENTGEGENGENOMMEN | M | M | ME → «266» |
| 12. | ENTGEGENGENOMMEN | EN → «256» | «256» | - |

▲ Cursor-Position

LZW-Transformation: ENTGEGENGENOMMEN wird zu ENTGE259N261OMM256

LZW-Code Dekomprimierung

| Zeichenkette, bestehend aus ASCII 0..255 oder Index ab 256 | | Erster Buchstabe in der Ausgabe | Buchstabe oder Wörterbuch | Vorherige Ausgabe und aktuelles Zeichen |
|--|---------------------------|---------------------------------|---------------------------|---|
| Schritt | Übermittelte Zeichenkette | Aktuelles Zeichen | Ausgabe | Temporärer Wörterbucheintrag |
| 1. | ENTGE«259»N«261»OMM«256» | E | E | - |
| 2. | ENTGE«259»N«261»OMM«256» | N | N | EN → «256» |
| 3. | ENTGE«259»N«261»OMM«256» | T | T | NT → «257» |
| 4. | ENTGE«259»N«261»OMM«256» | G | G | TG → «258» |
| 5. | ENTGE«259»N«261»OMM«256» | E | E | GE → «259» |
| 6. | ENTGE«259»N«261»OMM«256» | G | GE | EG → «260» |
| 7. | ENTGE«259»N«261»OMM«256» | N | N | GEN → «261» |
| 8. | ENTGE«259»N«261»OMM«256» | G | GEN | NG → «262» |
| 9. | ENTGE«259»N«261»OMM«256» | O | O | GENO → «263» |
| 10. | ENTGE«259»N«261»OMM«256» | M | M | OM → «264» |
| 11. | ENTGE«259»N«261»OMM«256» | M | M | MM → «265» |
| 12. | ENTGE«259»N«261»OMM«256» | E | EN | ME → «266» |

▲ Cursor-Position

LZW-Rücktransformation: ENTGE259N261OMM256 wird zu ENTGEGENGENOMMEN

Aufgaben zum LZW-Verfahren Zeit: 10 Min.

1. Erstellen sie die LZW-Codierung für das folgende Wort «ANANAS» und überprüfen sie mit der Dekodierung ihr Resultat:

ANANAS

2. Versuchen sie den erhaltenen LZW-Code zu dekomprimieren:

ERDBE<256>KL<260>



Aufgaben zum LZW-Verfahren Musterlösung

1. Erstellen sie die LZW-Codierung für das folgende Wort «ANANAS» und überprüfen sie mit der Dekodierung ihr Resultat:

ANANAS = AN«256»AS

2. Versuchen sie den erhaltenen LZW-Code zu dekomprimieren:

ERDBE<256>KL<260> = ERDBEERKLEE

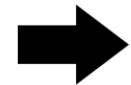


BWT (Burrows-Wheeler-Transformation)

Neben RLC, Huffman und LZW ist BWT eine weitere Art, wie Daten verlustlos komprimiert werden können.

BWT ist allerdings kein Algorithmus, der Daten direkt komprimiert. Vielmehr besteht seine Aufgabe darin, das Datenmaterial für eine anschliessende effektive Datenreduktion mit z.B. RLC vorzubereiten → **Präprozessor**.
(Hinweis: BWT führt nicht in allen Fällen zu einer vorteilhaften Zusammenführung von gleichen Buchstaben.)

1. Schritt:
ERDBEERE rotieren



2. Schritt:
Alphabetisch sortieren



3. Schritt:
Letzte Spalte und Position
des Originals werden übermittelt

ERDBEERE
EERDBEER
REERDSEE
EREERDBE
EEEREERDB
BEEREERD
DBEEREER
RDBEEERE

BEEREERD
DBEEREER
EERDSEE
EREERDB
EEEREERDB
BEEREERD
DBEEREER
RDBEEERE

1 : BEEREERD
2 : DBEEREER
3 : EERDSEE
4 : EEREERDB
5 : ERDBEEERE
6 : EREERDBE
7 : RDBEEERE
8 : REERDBEE

DRRBEEEE oder 1D2R1B4E

BWT (Burrows-Wheeler-Transformation)

Rücktransformation

Übermittelte Zeichenfolge ...

| CHAR | D | 2 | R | B | 4 | E | 5 |
|------|---|---|---|---|---|---|---|
|------|---|---|---|---|---|---|---|

Expandierte Zeichenfolge ...

| CHAR | D | R | R | B | E | E | E | E |
|------|---|---|---|---|---|---|---|---|
| POS | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Alphabetisch sortiert... (POS wird zu NEXT / Startzeichen an POS 5)

| POS | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|------|---|---|---|---|---|---|---|---|
| CHAR | B | D | E | E | E | E | R | R |
| NEXT | 4 | 1 | 5 | 6 | 7 | 8 | 2 | 3 |

Originaltext erstellen... (NEXT zeigt auf den nächsten Buchstaben)

An POS 5 steht der erster Buchstabe «E». Der nächste Buchstabe steht an POS 7...

E

An POS 7 steht der zweite Buchstabe «R». Der nächste Buchstabe steht an POS 2...

E R

An POS 2 steht der dritte Buchstabe «D». Der nächste Buchstabe steht an POS 1...

E R D

An POS 1 steht der vierte Buchstabe «B». Der nächste Buchstabe steht an POS 4...

E R D B

An POS 4 steht der fünfte Buchstabe «E». Der nächste Buchstabe steht an POS 6...

E R D B E

An POS 6 steht der sechste Buchstabe «E». Der nächste Buchstabe steht an POS 8...

E R D B E E

An POS 8 steht der siebte Buchstabe «R». Der nächste Buchstabe steht an POS 3...

E R D B E E R

An POS 3 steht der achte und letzte Buchstabe «E».

E R D B E E R E

Aufgaben zum BWT-Verfahren Zeit: 7 Min.

1. Erstellen sie die BWT-Transformation für das folgende Wort und überprüfen sie mit der Rücktransformation ihr Resultat:

ANANAS

2. Sie erhalten den folgenden Code in der Burrows-Wheeler-Transformation.
Welches Wort verbirgt sich dahinter?

IICRTGH6



Aufgaben zum BWT-Verfahren Musterlösung

1. Erstellen sie die BWT-Transformation für das folgende Wort und überprüfen sie mit der Rücktransformation ihr Resultat:

ANANAS = SNNAAA1

2. Sie erhalten den folgenden Code in der Burrows-Wheeler-Transformation.
Welches Wort verbirgt sich dahinter?

IICRTGH6 = RICHTIG



ZIP-Archiv

Das ZIP-Dateiformat (Reissverschluss) ist ein Format für verlustfrei komprimierte Dateien, das einerseits den Platzbedarf bei der Archivierung reduziert und andererseits als Containerdatei dient, in der mehrere zusammengehörige Dateien oder auch ganze Verzeichnisbäume zusammengefasst werden können.

Das Dateiformat und das ZIP-Kompressionsverfahren "Deflate" sind Public Domain.
ZIP sind: PKZIP, Info-ZIP, PeaZip, Xarchiver, 7-Zip, WinRAR, WinZip.

Übrigens: Die Deflate-Komprimierungsmethode findet sich in zahlreichen weiteren Formaten, wie z.B.

- Portable Network Graphics (PNG)
- Tagged Image File Format (TIFF)
- ISO-OpenDocumentformat
- ISO-Office-Open-XML-Format

Siehe auch:

<https://de.wikipedia.org/wiki/ZIP-Dateiformat>

<https://de.wikipedia.org/wiki/Deflate>

<https://de.wikipedia.org/wiki/Lempel-Ziv-Storer-Szymanski-Algorithmus>

<https://de.wikipedia.org/wiki/LZ77>

Aufgaben zu ZIP-Komprimierung Zeit: 15 Min. (Als Hausaufgabe zu beenden)

Es soll die Effizienz bei der ZIP-Komprimierung anhand von unterschiedlichen ASCII-Textdateien untersucht werden. Für diese Aufgaben wird ein Textgenerator (siehe Internet) empfohlen.

1. Erstellen sie die Datei D10.txt als ANSI-ASCII-Datei mit **10** zufällig gewählten Zeichen.
2. Nun D100.txt mit **100** zufällig gewählten Zeichen.
3. Danach D1000.txt mit **1'000** zufällig gewählten Zeichen.
4. Jetzt D10000.txt: mit **10'000** zufällig gewählten Zeichen.
5. Schlussendlich D100000.txt mit **100'000** zufällig gewählten Zeichen.
6. Erstellen sie mit D10.txt die ZIP-Datei Z10.zip
7. Nun mit D100.txt die ZIP-Datei Z100.zip
8. Danach mit D1000.txt die ZIP-Datei Z1000.zip
9. Jetzt mit D10000.txt die ZIP-Datei Z10000.zip
10. Schlussendlich mit D100000.txt die ZIP-Datei Z100000.zip
11. Erfassen sie nun in einer EXCEL-Tabelle die Speichergrößen aller ASCII-Dateien und den entsprechenden ZIP-Dateien. (Siehe Datei/Eigenschaften/Grösse (Nicht Grösse auf dem Datenträger!))
12. Wie interpretieren bzw. begründen sie ihr Resultat?
Tipp: Sie können in EXCEL Zahlenreihen auch grafisch anzeigen.
13. Erstellen sie eine ASCII-Textdatei mit **100'000 mal dem Buchstaben A** und zippen sie diese.
Vergleichen sie nun die beiden ZIP-Dateien mit 100'000 Zeichen. Wie erklären sie sich den Unterschied der Speichergrößen?
14. Was bewirkt ZIP, wenn die Originaldatei (z.B. diese beiden JPG-Bilder) bereits komprimiert vorliegen?

<https://www.juergarnold.ch/Kompression/ZIPTestHi.jpg>

<https://www.juergarnold.ch/Kompression/ZIPTestLo.jpg>



Aufgaben zu ZIP-Komprimierung

Musterlösungen

Es soll die Effizienz bei der ZIP-Komprimierung anhand von unterschiedlichen ASCII-Textdateien ermittelt werden:

ZIP-Komprimierung: (Alle Angaben in Byte)

ASCII-Textdateien (Mit repetitivem Inhalt ABCDEFGHIJKLMNOP...)

Test10 : 10 / 128

Test100 : 100 / 135

Test1000 : 1'000 / 143

Test10000 : 10'000 / 172

Test100000 : 100'000 / 348

Random100000 : 100'000 / 4141 (Zufallsabfolge)

CHAR100000 : 100'000 / 241 (Alles Buchstabe A)

Bilddateien:

Original: 1000 x 1000 x 3 = 3'000'000

TestLo.jpg : 32810

TestLo.zip : 17590

TestHi.jpg : 271201

TestHi.zip : 243119

