LB1

Informationsspeicher

Titel	Information
ZIP	**Kompressionsalgorithmus** - Kombination aus mehreren Algorithmen, wie *Huffman*, LZ77 und *BWT*
ASCII Codierung	A-Z = 65-90, a-z = 97-122

Codierung

Zahlensysteme

Von	Zu	Formel	Bemerkung	Bsp. von	Bsp. zu	Beispiel
Dez	Hex	Hex = Dez ÷ 16	Rest als Hex-Ziffer aufschreiben bis Quotient 0 ist.	75	4B	75 ÷ 16 = `4` Rest `11 = B`
Dez	Bin	Bin = Dez ÷ 2	Rest aufschreiben, bis Quotient 0 ist, dann umdrehen	13	1101	13 ÷ 2 = 6 Rest `1`, 6 ÷ 2 = 3 Rest `0`, 3 ÷ 2 = 1 Rest `1`, 1 ÷ 2 = 0 Rest `1`
Hex	Dez	Dez = ∑(Ziffer × 16^{pos})	POS = Position von rechts	1F	31	$1 \times 16^1 + 15 \times 16^0$
Hex	Bin	Bin = f	f = Jede Hex-Ziffe in ihre 4-Bit-Binärform umwandeln	2A	0010 1010	2 = 0010, A = 10 = 1010
Bin	Dez	Dez = ∑(Bit × 2^{pos})	POS = Position von rechts	1011	11	$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
Bin	Hex	Hex = f	f = Binärzahl in 4er-Gruppen von rechts aufteilen und in Hex umwandeln	101110	2E	10 1110 -> 1110 = 14 = E, 10 = 2 -> 2E

Bits

Bemerkung	Formel	X	Beispiel
Y Bits benötigt für Dezimal **X**	Y = log(X) ÷ log(2)	1000	**Y** = log(1000) ÷ log(2) = **10 Bit**
X Bits ergeben **Y** Kombinationen	$Y = 2^x$	16	**Y** = 2^{16} = **65'536 Kombinationen**
XMHz = **X**Hz × 10^6 -> **X**Hz = **X** Bit/s (Wenn ein Takt = 1 Bit überträgt)	-	1	1MHz = 1Hz × 10^6 = 125kB/s
Seriell **X** vs. **Y**-Bit-parallel -> **Z** Bit/s	Z = X × Y	X = 100MHz, Y = 8 Bit	100MHz × 8 Bit = 800 MBit/s

Wertebereich Integer

Y = Max. Wert

Vorzeichen	Grösse in Bit (X)	Min	Max	Formel	Beispiel
Unigned	32-Bit	00000000 (0)	11111111 (255)	$Y = 2^x$	2^{32}
Signed	32-Bit	10000000 (-128)	01111111	$Y = 2^x \div 2$ -	$2^{32} \div 2$ - 1

Vorzeichen	Grösse in Bit (X)	Min	(127) Max	1 Formel	Beispiel

Zweierkomplement-Bildung

Wert	1. In Binär umwandeln	2. Bits invertieren *(1 -> 0, 0 -> 1)*	3. 1 addieren
+83	01010011	10101100	10101101 **(-83)**

Fliesskommazahlen (Float)

Norm IEEE 754: $x = v * m * b^e$

Type	Vorzeichen v	Exponent e	Mantisse m	Basis b (bei normalisierten Gleitkommazahlen 2)
Single	1 Bit	8 Bit	23 Bit	2
Double	1 Bit	11 Bit	52 Bit	2

Binäres Rechnen und Data-Overflow

- **Beispiel: 78** (01001110) + **167** (10100111) = **245** (11110101)
- **Beispiel Data-Overflow: 200** (11001000) + **180** (10110100) = **380** (**1**01111100) -> Data-Overflow -> **380** - **256** = **124** (01111100)

Kompression

Verfahren	Abkürzung	Typ
Variable-Length Code	VLC	Kodierungsverfahren
Huffman-Codierung	Huffman	Kodierungsverfahren
Run-Length Encoding	RLE	Kompressionsmethode
Lempel-Ziv-Welch	LZW	Kompressionsmethode
Burrows-Wheeler-Transformation	BWT	Vorverarbeitung
ZIP	ZIP	Kompressionsalgorithmus

VLC Beispiel: BANANA

1. Zeichenhäufigkeit

Zeichen	Häufigkeit
B	1
A	3
N	2

2. Kürzere Code für häufige Zeichen zuweisen

Zeichen	Feste Länge (ASCII)	Variable Länge (VLC)
B	01000010 (8 Bit)	00 (2 Bit)
A	01000001 (8 Bit)	1 (1 Bit)
N	01001110 (8 Bit)	01 (2 Bit)

3. Wort kodieren

```
B(00) A(1) N(01) A(1) N(01) A(1)
= 00 1 01 1 01 1
= 00101101 1
```

4. Vergeich mit fester Länge

- **ASCII (fest Länge, 8 Bit pro Zeichen)** -> `BANANA` = `6 * 8 = 48 Bit`
- **VLC (variable Länge, durchschnittlich ~2 Bit pro Zeichen)** -> `BANANA` ≈ **10 Bit**

Ersparnis: ~80% Speicherplatz

Huffman Beispiel: HEDGEHOG

1. Buchstabenhäufigkeit:

Char	Count
D	1
O	1
H	2
E	2
G	2

2. Binary-Tree

2.1 Node für jedes Zeichen (key) und Häufigkeit (value):

```
(D, 1)  (O, 1) (H, 2) (E, 2) (G, 2)
```

2.2 Nodes mit geringsten Häufigkeit verBinden zu neuer Node mit der Summe der Child-Nodes:

```
    (DO,2)
    /   \
(D, 1)  (O, 1) (H, 2) (E, 2) (G, 2)
```

2.3 Wiederholen, bis nur noch eine Node übrig ist (root).

```
              (DOHEG|8)
              /   \
          (DOH|4)     (EG|4)
       / \           / \
    (DO|2) (H|2)    (E|2) (G|2)
    /  \
(D|1)  (O|1)
```

3. Codes zuweisen

```
            (DOHEG|8)
          1 /      \ 0
      (DOH|4)      (EG|4)
      1 / \ 0       1 / \ 0
    (DO|2) (H|2)   (E|2)  (G|2)
  1 / \ 0
(D|1)  (O|1)

D = 111
O = 110
H = 10
E = 01
G = 00

# Koordinaten
D(000) O(001) H(01) E(10) G(11)
= 000 0001 01 10 11
= 0000001011011
```

RLE Beispiel: AAAABBBCCDAA

1. Zeichenfolgen ersetzen:

Original	Gekürzt
AAAA	4A
BBB	3B
CC	2C
D	1D
AA	2A

Ergebnis: 4A3B2C1D2A

LZW Beispiel: ABABABAABABA

1. Initialisierung, ASCII Code basiertes Wörterbuch:

Zeichen	Code
A	65
B	66

2. Codierung:

Schritt	Eingabe	Wörterbuch vorhanden?	Ausgabe	Neuer Index
1	A	Ja	65	-
2	B	Ja	66	-
3	AB	Nein	-	`256 = AB`
4	A	Ja	65	-
5	BA	Nein	-	`257 = BA`
6	B	Ja	66	-
7	AB	Ja	256	-

8	ABA	Nein	-	`258 = ABA`
Schritt 9	Eingabe B	Wörterbuch vorhanden?	Ausgabe 66	Neuer Index -
10	AB	Ja	256	-
11	ABA	Ja	258	-
12	ABAB	Nein	-	`259 = ABAB`

3. Kompreimierte Ausgabe: `[65,66, 65, 66, 256, 66, 256, 258]`

BWT Beispiel: ANANAS

1. Transformation

1.1 Rotation

	1	2	3	4	5	6
1	**A**	N	A	N	A	S
2	S	**A**	N	A	N	A
3	A	S	**A**	N	A	N
4	N	A	S	**A**	N	A
5	A	N	A	S	**A**	N
6	N	A	N	A	S	**A**

1.2 Alphabetische Sortierung

	1	2	3	4	5	6
1	**A**	N	A	N	A	S
2	A	N	A	S	**A**	N
3	A	S	**A**	N	A	N
4	N	A	N	A	S	**A**
5	N	A	S	**A**	N	A
6	S	**A**	N	A	N	A

1.3 Letzte Spalte und Index von Original Ausgeben

	1	2	3	4	5	6
1	A	N	A	N	A	**S**
2	A	N	A	S	A	**N**
3	A	S	A	N	A	**N**
4	N	A	N	A	S	**A**
5	N	A	S	A	N	**A**
6	S	A	N	A	N	**A**

Ausgabe: SNNAAA1

Nach *RLE*: 1S2N3A1

2. BW-Rücktransformation

2.1 Übermittelte Zeichenfolge: *S 2N 3A 1*

2.2 Expandierte Zeichenfolge:

```
CHAR | S N N A A A
POS  | 1 2 3 4 5 6
```

2.3 Alphabetisch sortiert (POS wird zu NEXT / Startzeichen an POS 1):

```
POS  | 1 2 3 4 5 6
CHAR | A A A N N S
NEXT | 4 5 6 2 3 1
```

2.4 Originaltext erstellen:

- Erster Buchstabe an POS 1 (Letzte Ziffer an input)
- POS von nächstem Zeichen = *NEXT*

0	1	2	3	4	5	6
1	**A**					
2	A	**N**				
3	A	N	**A**			
4	A	N	A	**N**		
5	A	N	A	N	**A**	
6	A	N	A	N	A	**S**

Verlustlose vs. verlustbehaftete Kompression

Art	Eigenschaft	Anwendung	Verfahren
Verlustlos	Daten können 100% identisch wiederhergestellt werden.	Text, Applikationen, Datenbanken	VLC, Huffman, RLC, LZ77/LZ78, LZW, *BWT*
Verlustbehaftet	Entfernen nicht wahrnehmbare Details	JPEG, MP3, MP4	MPEG