LZW-Kompression

Die LZW-Kompression ist ein **Wörterbuchverfahren** nach Lempel-Ziv-Welch.

Wörterbuchverfahren hinterlegen **wiederkehrende Zeichenfolgen** in einem **Wörterbuch**. Kommen diese Zeichenfolgen dann im zu komprimierenden Text erneut vor, reicht ein Verweis auf diesen Eintrag. Das LZW-Verfahren arbeitet dabei mit einem dynamischen Wörterbuch, welches direkt während der Kompression selbst erzeugt wird und damit keinen zusätzlichen Speicherplatz benötigt.

Um Platz für das Wörterbuch neben den normalen (ASCII-)Zeichen zu schaffen, reichen 8 Bit nicht aus. Für gewöhnlich werden 12 Bit für jedes Zeichen bzw. jeden Wörterbucheintrag verwendet. Das Wörterbuch kann also maximal 2^{12} = 4096 Zeichen und Zeichenkombinationen beinhalten, wovon die ersten 256 Einträge bei Texten fest mit den ASCII-Zeichen vorbelegt sind.

Die Codierung verläuft nach folgendem **Algorithmus**:

1. Lies eine **möglichst lange** Zeichenkette ein, die bereits im Wörterbuch steht. Zu Beginn ist das jeweils nur ein einzelnes Zeichen!
2. **Schreibe** den Code des **gefundenen Eintrags** in die **Ausgabe**.
3. Lege aus der eben gefundenen Zeichenkette und dem **nachfolgenden** Zeichen einen neuen Wörterbucheintrag mit der nächst möglichen Codierung an.
4. Wenn nötig wird das letzte Byte der Ausgabe mit 0 aufgefüllt

Beispiel

Codierung

Die Zeichenkette BABAABBAA soll mit LZW **codiert** werden. Das Wörterbuch ist zu Beginn des Vorgangs im Bereich von 000_{16} bis $0FF_{16}$ mit den ASCII-Zeichen befüllt[1)]. Zum besseren Verständnis des weiteren Ablaufs sollte man im Hinterkopf haben, dass der ASCII Code des großen A $65_{10}=41_{16}$ ist, der des großen B $66_{10}=42_{16}$

Dez	Hex	Sym
64	40	@
65	41	A
66	42	B
67	43	C

Noch zu bearbeitende Zeichenkette	Gefundener Eintrag	Ausgabe (12Bit)	Neuer Wörterbucheintrag
BABAABBAA	B ← 042_{16}	042_{16}	BA → 100_{16}
ABAABBAA	A ← 041_{16}	041_{16}	AB → 101_{16}
BAABBAA	BA ← 100_{16}	100_{16}	BAA → 102_{16}
ABBAA	AB ← 101_{16}	101_{16}	ABB → 103_{16}

Noch zu bearbeitende Zeichenkette	Gefundener Eintrag	Ausgabe (12Bit)	Neuer Wörterbucheintrag
BAA	BAA ←102_{16}	102_{16}	

Die Zeichenfolge wird also folgendermaßen codiert: $\mathbf{042041100101102_{16}}$. Das sind 7,5 Bytes [2]. Die Kompression ist also bei solch kurzen Zeichenketten noch nicht drastisch - wenn man sich jedoch vorstellt, dass das Wörterbuch stets längere Zeichenketten mit einem einzigen 12Bit Code zugreifbar macht, kann die Kompression unter Umständen bei längeren Texten deutlich stärker ins Gewicht fallen.

Decodierung

Bei der **Decodierung** werden 12-Bit-Blöcke eingelesen. Das Wörterbuch wird während des Vorgangs mit Einträgen befüllt die aus dem ersten Zeichen des aktuellen Eintrag und dem vorangehenden Eintrag bestehen. Wir nehmen den codierten String von oben: $\mathbf{042041100101102_{16}}$.

Aktueller 12Bit-Block (Hexadezimal)	Gefundener Eintrag (erster Buchstabe)	Neuer Wörterbucheintrag	Ausgabe
042	B (B)		B
041	A (A)	BA = 100_{16}	A
100	BA (B)	AB = 101_{16}	BA
101	AB (A)	BAA = 102_{16}	AB
102	BAA (B)	ABB = 103_{16}	BAA

Decodiert lautet der Text also BABAABBAA.

> **Anmerkung**: Das Wörterbuch musste zur Decodierung hier **nicht** gesondert übertragen werden - es "ensteht" während des Vorgangs.

Für die Aufgaben kannst du die folgenden Arbeitsblätter verwenden:

- Vorlage: Codierung (Erklärung in diesem Video)
- Vorlage: Decodierung (Erklärung in diesem Video)

(A1)

Codiere den Text ABABCABCDABCD und vergleiche die codierte und die uncodierte Länge miteinander.

Lösung:

- Codiert: 041042100043102044104
- Der uncodierte Text war 13 Zeichen = 13 Bytes lang
- Die Codierung benötigt 7*12 Bit = 10,5 Bytes lang, was am Ende 11 Bytes belegt.

(A2)

Decodiere folgenden Code: `058059060101100103`. Die ASCII-Tabelle findest du hier

Lösung:

Daraus wird der Text: XYZYZXYYZX

(A3)

Versuche, den Code `042041100101041104` zu decodieren. Welches Problem ergibt sich dabei?

Lösung:

Im letzten Schritt wird auf den Eintrag `104` verwiesen, der bei der Decodierung jedoch noch nicht existiert. Das ist ein Sonderfall, der auftritt, wenn eine Zeichenfolge mehrfach direkt hintereinander vorkommt. Dann gilt: der „gefundene“ Eintrag entspricht dem vorherigen Eintrag + dem ersten Buchstaben des vorherigen Eintrags.

(A4)

Der folgende LZW-Code: 0 1 2 4 6 5 7 7 3 codiert eine Pixelgrafik, die 4 Pixel breit ist. Die einzelnen auftretenden Pixel haben den folgenden "Grundcode":

(A5)

Erläutere in einem kurzen Text das Grundprinzip der Komprimierung beim LZW-Verfahren.

(A6)

Begründe, dass das LZW-Verfahren nicht jede Eingabe komprimieren kann.

Lösung

Das LZW-Verfahren ist ein verlustfreies Verfahren, d.h. jede Eingabe ist eindeutig wiederherstellbar. Zu jeder komprimierten Bitfolge gehört damit genau eine Eingabe. Es kann kein verlustfreies Verfahren geben, das jede Eingabe komprimiert.

Begründung: Wenn es ein Verfahren gäbe, das jede Eingabe verkürzen kann, könnte man dieses wieder- holt anwenden, bis die Ausgabe nur noch 1 Bit lang wäre. Diese könnte genau zwei Werte annehmen, 0 oder 1. Daraus könnte man aber höchstens zwei Eingaben rekonstruieren.

Material

01_lzw-vorlage-codierung.odt	479.1 KiB	03.10.2022	16:53
01_lzw-vorlage-codierung.pdf	74.6 KiB	03.10.2022	16:53
01_lzw-vorlage-decodierung.odt	478.4 KiB	03.10.2022	16:53
01_lzw-vorlage-decodierung.pdf	73.4 KiB	03.10.2022	16:53
06-kompression-lzw.odp	43.0 KiB	29.09.2022	07:20
06-kompression-lzw.pdf	157.1 KiB	29.09.2022	07:20
ab.png	9.9 KiB	03.10.2022	15:58
lzw-codierung-beispiel.png	130.2 KiB	14.01.2025	09:15
lzw-decodierung-beispiel.png	94.9 KiB	14.01.2025	09:06
lzw_a2_cod_dec.odp	78.4 KiB	05.12.2023	14:01
lzw_a2_cod_dec.pdf	81.5 KiB	05.12.2023	14:01

pixel.png	2.0 KiB	03.10.2022 17:01

[1]

Die ersten 256 Zeichen des 12 Bit Raums, der für die Codierung zur Verfügung steht

[2]

wenn man mit ganzen Bytes arbeiten möchte, wird das mit Nullen zu 8 Byte aufgefüllt

From:
https://info-bw.de/ -

Permanent link:
https://info-bw.de/faecher:informatik:oberstufe:codierung:lzw:start?rev=1701706658

Last update: **04.12.2023 16:17**