# Lempel-Ziv-Welch (LZW) Kompression

## 1. Einleitung

Das im Bereich der Computergraphik weit verbreitete Kompressionsverfahren ist eine verlustfreie Datenkompressionsmethode, die in den verschiedensten Grafikformaten, wie GIF und TIFF Verwendung findet. Sie ist auch Teil des V.42- Modem Standards und wird auch in PostScript Level 2 Dateien verwendet.

Im Jahre 1997 schufen Abraham Lempel und Jakob Ziv den ersten Kompressions-algorithmus der "LZ-Familie". Dieser "LZ77" Algorithmus findet sich in vielen Textverarbeitungsprogrammen sowie in verschiedenen Kompressionsprogrammen. Der modifizierte "LZ78" Algorithmus wird meist für Binärdateien und Bitmaps verwendet. Im Jahre 1984 modifizierte Terry Welch den LZ78, um ihn in "high-performance disc controller" einzubauen. Dies ist der "LZW" Algorithmus, wie er heute eingesetzt wird.

Der "LZW" Algorithmus funktioniert mit fast allen Sorten von Ausgangsdaten und ist im allgemeinen sehr schnell, sowohl bei der Kompression als auch bei der Dekompression und benötigt keine Gleitkommaarithmetik. Nachdem der LZW seine komprimierten Daten als Bytes und nicht als Worte ablegt, ist er von der Byteanordung der verschiedenen Plattformen unabhängig. Der LZW wird als Substitions- oder wörterbuchbasierender Algorithmus bezeichnet. Er erstellt ein Wörterbuch (das auch Übersetzungstabelle oder Stringtabelle genannt wird) aus den unkomprimierten Daten. Dazu wird der unkomprimierte Datenstrom in einzelne Zeichenketten zerlegt, die jeweils mit den schon vorhandenen Wörterbucheinträgen verglichen werden. Findet sich bereits ein Eintrag, so wird im komprimierten Ausgabestrom nur mehr die Kennung des Wörterbucheintrages angegeben. Findet sich kein Eintrag im Wörterbuch, so wird ein neuer Einrag erstellt, in der Hoffnung, ihn später wieder benutzen zu können.

## 2. Kodieren

LZW gibt nur Kodes und keine Zeichen aus. Dies setzt voraus, dass zu Beginn der Kodierung/Dekodierung für jedes im Eingabealphabet vorkommende Zeichen ein entsprechender Eintrag im Wörterbuch existiert. Da deshalb jede Kodierung mit einem Präfix der Länge "1" beginnt, sucht der Kodierer im Wörterbuch immer zuerst nach Zeichenketten der Länge "2". Ausserdem muss das erste Zeichen des neuen Präfix mit dem letzten Zeichen der zuvor eingetragenen Zeichenkette identisch sein, da dem Dekodierer sonst dieses Zeichen fehlt, und er so das Wörterbuch nicht korrekt aufbauen könnte.

### 2.1 Der Algorithmus in Einzelschritten

1. Zu Beginn ist der Präfix leer, das Wörterbuch enthält für jedes im Dateneingabestrom vorkommendes Zeichen einen Eintrag.
2. z := nächstes Zeichen aus dem Eingabedatenstrom
3. Ist "Präfix + z" im Wörterbuch ?
   - JA: Präfix := "Präfix + z"
   - NEIN:
     1. Gebe den Kode für "Präfix " aus.
     2. Trage "Präfix + z" im Wörterbuch ein.
     3. Präfix := z
4. Ist das Ende des Eingabedatenstroms erreicht ?
   - NEIN: Gehe zu Schritt 2.
   - JA: Wenn Präfix nicht leer ist, gebe seinen korrespondierenden Kode aus.

### 2.2 Beispiel

| Eingabe | Ausgabe | Präfix | J/N | Wörterbuch | |
|---|---|---|---|---|---|
| A B B A B A B A C | - | - | | 1<br>2<br>3 | A<br>B<br>C |
| A | - | A | J | | |
| B | 1 | B | N | 4 | AB |
| B | 2 | B | N | 5 | AB |
| A | 2 | A | N | 6 | BA |
| B | - | AB | J | 7 | ABA |
| A | 4 | A | N | | |
| B | - | AB | J | | |
| A | - | ABA | J | | |
| C | 7 | C | N | 8 | ABAC |
| | 3 | - | - | | |

**-> 122473**

## 3. Dekodieren

Da bei der LZW-Kodierung keine Zeichen explizit ausgegeben werden dürfen, musste der LZW-Kode so konzipiert werden, dass der Dekodierer immer noch die Möglichkeit hat, das Wörterbuch korrekt aufzubauen. Dieses Problem wurde mit der Vereinbarung gelöst, dass das erste Zeichen eines neuen Wörterbucheintrages identisch sein muss mit dem letzten Zeichen des Vorhergehenden. Dadurch hinkt der Dekodierer dem Kodierer bei der Erstellung des Wörterbuches allerdings einen Eintrag hinterher, da er einen neuen Eintrag erst dann vornehmen kann,wenn er den nächsten Kode übersetzt hat, weil er ja dessen erstes Zeichen benötigt.

### 3.1 Der Algorithmus in Einzelschritten

1. Das Wörterbuch enthält zu Beginn für jedes im Eingabedatenstrom vorkommende Zeichen einen Eintrag.
2. K := erster Kode aus dem Eingabedatenstrom (bez. immer ein Zeichen)
3. Gib "K" aus.
4. merke K in AK.
5. K := nächster Kode im Eingabedatenstrom
6. Ist "K" im Wörterbuch ?
   - JA:
     1. Gib "K" aus.
     2. Präfix := "AK"
     3. z := erstes Zeichen von "K"
     4. Trage "Präfix + z" in das Wörterbuch ein.
   - NEIN:
     1. Präfix := "AK"
     2. z := erstes Zeichen von "AK"
     3. Trage "Präfix + z" (="K"!) in das Wörterbuch ein UND gib es aus.
7. Ist das Ende des Eingabedatenstroms noch nicht erreicht, dann gehe zu Schritt 4.

### 3.2 Beispiel

| Eingabe | Ausgabe | AK | Z | Präfix | J/N | Wörterbuch | |
|---|---|---|---|---|---|---|---|
| 1 2 2 4 7 3 | - | - | - | - | | 1<br>2<br>3 | A<br>B<br>C |
| 1 | A | A | - | - | - | | |
| 2 | B | | B | A | J | 4 | AB |
| 2 | B | B | B | B | J | 5 | BB |
| 4 | AB | B | A | B | J | 6 | BA |
| 7 | ABA | AB | A | AB | N | 7 | ABA |
| 3 | C | ABA | C | ABA | | 8 | ABAC |

**-> ABBABABAC**