Kompressionsverfahren: Übersicht

- Klassifikationen:
  - Universell vs. speziell (für bestimmte Informationstypen)
  - Verlustfrei vs. verlustbehaftet
  - In diesem Kapitel: nur universelle & verlustfreie Verfahren

- Im folgenden vorgestellte Verfahren:
  - Statistische Verfahren:
    - » Huffman-Codierung
    - » Arithmetische Codierung
  - Zeichenorientierte Verfahren:
    - » Lauflängencodierung (RLE Run Length Encoding)
    - » LZW-Codierung 

LZW-Decodierung bei bekannter Tabelle

Wiederhole solange Eingabe nicht leer:

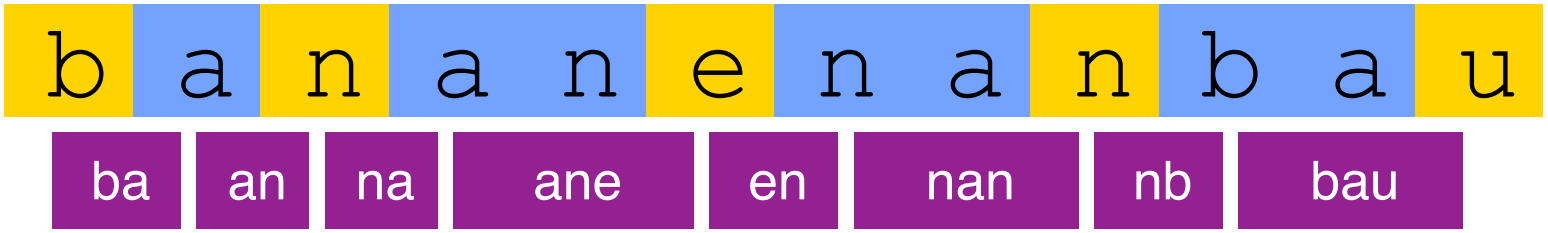
k = NächsteEingabezahl;

Schreibe Zeichenreihe mit Tabellenindex k auf Ausgabe;

Ende Wiederholung;

LZW-Decodierung (1)

- Grundidee („symmetrische Codierung“):
  - Das aufgebaute Wörterbuch muß *nicht* zum Empfänger übertragen werden.
  - Das Wörterbuch wird nach dem gleichen Prinzip wie bei der Codierung bei der Decodierung dynamisch aufgebaut.
  - Das funktioniert, weil bei der Codierung immer *zuerst* der neue Eintrag für das Wörterbuch nach bekannten Regeln aus dem schon gelesenen Text aufgebaut wird, bevor der neue Eintrag in der Ausgabe verwendet wird.
- Algorithmusidee:
  - Neu einzutragendes Teilwort = letztes Teilwort plus erstes Zeichen des aktuellen Teilworts



LZW-Decodierung (2)

Prinzipieller Algorithmus:

SeqChar *p* := <>;
int *k* = NächsteEingabezahl;
Schreibe Zeichenreihe mit Tabellenindex *k* auf Ausgabe;
int *old* = *k*;
Wiederhole solange Eingabe nicht leer:
    k = NächsteEingabezahl;
    SeqChar *akt* = Zeichenreihe mit Tabellenindex *k*;
    Schreibe Zeichenreihe *akt* auf Ausgabe;
    p = Zeichenreihe mit Tabellenindex *old* (letztes Teilwort);
    Char *q* = erstes Zeichen von *akt*;
    Trage *p* & <*q*> in Tabelle ein
    (und erzeuge neuen Index dafür);
    old = *k*;
Ende Wiederholung;

LZW-Decodierung (3)

Beispielzeichenreihe:
“98-97-110-257-101-
258-110-256-117"

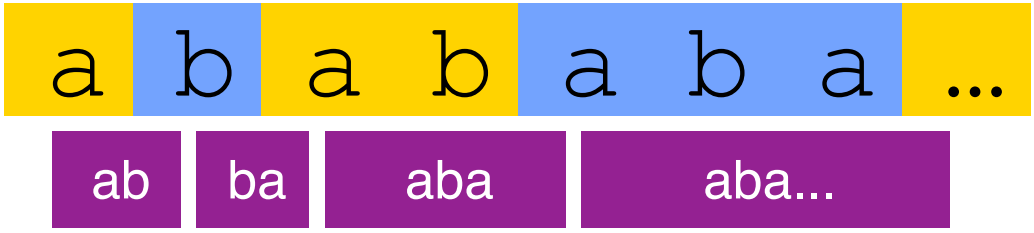
Wiederhole solange Eingabe nicht leer:
k = NächsteEingabezahl;
SeqChar *akt* = Zeichenreihe mit Tabellenindex *k*;
Schreibe Zeichenreihe *akt* auf Ausgabe;
p = Zeichenreihe mit Tabellenindex *old* (letztes Teilwort);
Char *q* = erstes Zeichen von *akt*;
Trage *p* & <*q*> in Tabelle ein
(und erzeuge neuen Index dafür);
old = *k*;
Ende Wiederholung;

Lesen (k)	**Ausgabe** (q ist jeweils unterstrichen)	**Puffer füllen (p)**	**Codetabelle schreiben (p & <q>)**	**Merken (old)**
98	b			98
97	**a**	b	(<ba>, 256)	97
110	**n**	a	(<an>, 257)	110
257	**a**n	n	(<na>, 258)	257
101	**e**	an	(<ane>, 259)	101
258	**n**a	e	(<en>, 260)	258
110	**n**	na	(<nan>, 261)	110
256	**b**a	n	(<nb>, 262)	256
117	**u**	ba	(<bau>, 263)	117
EOF				

LZW-Decodierung (4)

- Beispielzeichenreihe: “abababa...“, Beispielcode: “97-98-256-258“
- Ablauf:

Lesen (k)	Ausgabe (q ist jeweils unterstrichen)	Puffer füllen (p)	Codetabelle schreiben (p & <q>)	Merken (old)
97	a			97
98	**b**	a	(<ab>, 256)	98
256	**a**b	b	(<ba>, 257)	256
258	**???**			



Decodierung ist so noch nicht korrekt!

LZW-Decodierung, vollständige Fassung

SeqChar *p* := <>;
int *k* = NächsteEingabezahl;
Schreibe Zeichenreihe mit Tabellenindex *k* auf Ausgabe;
int *old* = *k*;
Wiederhole solange Eingabe nicht leer:
    k = NächsteEingabezahl;
    SeqChar *akt* = Zeichenreihe mit Tabellenindex *k*;
    p = Zeichenreihe mit Tabellenindex *old* (letztes Teilwort);
    Falls Index *k* in Tabelle enthalten
        dann **Char** *q* = erstes Zeichen von *akt*;
            Schreibe Zeichenreihe *akt* auf Ausgabe;
        sonst **Char** *q* = erstes Zeichen von *p*;
            Schreibe Zeichenreihe *p* & <*q*> auf Ausgabe;
    Ende Fallunterscheidung;
    Trage *p* & <*q*> in Tabelle ein
    (und erzeuge neuen Index dafür);
    old = *k*;
Ende Wiederholung;