Datenkompression

Sommersemester 2019 **Übungsblatt 3** Prof. Dr. Enno Ohlebusch

Institut für Theoretische Informatik Uwe Baier

Ausgegeben am 28.05.2019 Besprechung am 04.06.2019

Aufgabe 3.1

Die Zeichenfolge $S :=$ `MISS MISSION MISSISSIPPI` über dem Quellenalphabet $\Sigma := \{\texttt{M}, \texttt{I}, \texttt{S}, \texttt{O}, \texttt{P}, \texttt{N}\}$ soll komprimiert werden. Die Leerzeichen dienen dabei nur der Formatierung und sollen nicht mitkodiert werden.

a) Verwenden Sie das LZ77-Verfahren (Suchpuffer 7 Zeichen und Codierpuffer 3 Zeichen lang).

b) Verwenden Sie das LZSS-Verfahren (Suchpuffer 7 Zeichen und Codierpuffer 3 Zeichen lang).

c) Verwenden Sie das LZ78-Verfahren.

Aufgabe 3.2

Wir betrachten einen LZW-Dekodierer mit folgendem Anfangswörterbuch:

Index	Eintrag
1	`a`
2	`b`
3	`c`

Dekodieren Sie die Folge 1,2,3,6,4,8 mit dem angegebenen LZW-Dekodierer.

Aufgabe 3.3

In dieser Aufgabe befassen wir uns mit Grammatik-basierter Datenkompression.

a) Ein *Straight Line Program* ist eine Grammatik G, deren Sprache $L(G)$ nur aus genau einem Wort $w \in \Sigma^*$ besteht und die Produktionsregeln nur die folgende Form besitzen:

$$S \to AB$$
$$S \to a$$

Hierbei stehen S, A und B für beliebige Nichtterminalsymbole und a für ein beliebiges Terminalsymbol. Geben Sie ein Straight Line Program mit höchstens 8 Nichtterminalsymbolen sowie dessen Ableitungsbaum für das Wort $w =$ `textextext` an.

b) Das Kompressionsprogramm RePair [1] arbeitet wie folgt: Gegeben sei ein Wort w, dann sucht RePair nach dem am häufigsten in w auftretenden Buchstabenpaar xy, ersetzt alle (nicht-überlappenden) Vorkommen des Paares xy durch einen neuen Buchstaben X, und speichert sich die Regel $X \to xy$ ab. Dieser Prozess wird nun solange wiederholt, bis das Restwort nur noch Buchstabenpaare mit Häufigkeit 1 besitzt.

Geben Sie die Regeln und das Restwort nach der Anwendung von RePair auf das Wort $w =$ `textextext` an.

c) Beschreiben Sie, wie mithilfe von RePair für ein gegebenes Wort w ein Straight Line Program erzeugt werden kann.

Aufgabe 3.4

Schreiben Sie ein Programm, das als Eingabe eine Datei sowie eine Fenstergröße w enthält (diese dient sowohl für den Such- als auch für den Codierpuffer, siehe unten), und als Ausgabe die Komprimierungsgröße in Bits der Lempel Ziv-Kodierung in der Storer–Szymanski Variante ausgibt. Es gelten dabei folgende Rahmenbedingungen:

- Ist die längste im Fenster gelegene Wiederholung kleiner als 2 Zeichen lang, so soll der Buchstabe in die Kodierung aufgenommen werden. Dafür werden also

$$\underbrace{1}_{\text{Indikatorbit}} + \underbrace{8}_{\text{Buchstabe}} = 9 \text{ bits}$$

  benötigt.

- Ist die längste im Fenster gelegene Wiederholung 2 oder mehr Zeichen lang, so soll die Wiederholung entsprechend der Storer-Szymanski Variante aufgenommen werden. Weiter sind zwar überlappende Wiederholungen erlaubt, jedoch darf die Wiederholung nicht länger als $w + 1$ Zeichen sein.

  Dass bedeutet also, das man für die Kodierung der Wiederholung genau

$$\underbrace{1}_{\text{Indikatorbit}} + \underbrace{\lceil \log_2(w) \rceil}_{\text{Offset}} + \underbrace{\lceil \log_2(w) \rceil}_{\text{Länge}} = 1 + 2 \cdot \lceil \log_2(w) \rceil \text{ bits}$$

  benötigt: Der Offset besteht aus Zahlen zwischen 1 und w, d.h. wir können vom Offset eins abziehen und benötigen so die Bitbreite $\lceil \log_2(w) \rceil$. Weiter besteht die Länge aus Zahlen zwischen 2 und $w + 1$, d.h. wir können von den Länge immer den Wert 2 abziehen, um so die Bitbreite auf den gleichen Wert zu verringern.

Testen Sie mit ihrem Programm, wieviele Bits mindestens benötigt werden, um den String `textitext` zu kodieren.

Literatur

[1] N. Jesper Larsson and Alistair Moffat. Offline Dictionary-Based Compression. In *Proceedings of the Conference on Data Compression*, DCC '99, 1999.