Datenkompression

Sommersemester 2016 — **Übungsblatt 1** — Prof. Dr. E. Ohlebusch

Institut für Theoretische Informatik — J. Lorenz

Ausgegeben am 22.04.2016 — Besprechung am 29.04.2016

Aufgabe 1.1

Geben Sie jeweils einen Code an, der die folgenden Eigenschaften erfüllt.

a) Einen Code, der die Kraft-McMillan Ungleichung erfüllt.

b) Einen Code, der die Kraft-McMillan Ungleichung verletzt.

c) Einen eindeutig dekodierbaren Code, der weder Präfix- noch Suffixcode ist. Wobei Suffixcode folgendermaßen definiert werden kann: Wenn für einen Code C gilt, dass alle Wörter $a \in C$ **nicht** Suffix eines anderen Wortes $b \in C$ sind, dann nennen wir C einen Suffixcode.

Aufgabe 1.2

Auf dem Alphabet $\{d, i, l, m, n, r, u\}$ sei eine Wahrscheinlichkeitsfunktion P definiert:

x	`d`	`i`	`l`	`m`	`n`	`r`	`u`
$P(x)$	1/24	1/24	1/6	7/24	1/12	1/24	1/3

a) Bestimmen Sie einen Präfixcode nach dem Shannon-Verfahren.

b) Bestimmen Sie einen Präfixcode nach dem Shannon-Fano-Verfahren (Variante 2 in Aufgabe 1.3).

c) Bestimmen Sie einen Präfixcode nach dem Huffman-Verfahren.

d) Berechnen Sie die erwarteten Codewortlängen der Codes aus den Teilaufgaben a) bis c).

Aufgabe 1.3

Es gibt zwei Varianten des Shannon-Fano-Algorithmus. In der ersten Version wird die Liste der Symbole in zwei Teile $M_1 = (c_1, \ldots, c_k)$ und $M_2 = (c_{k+1}, \ldots, c_n)$ gespalten, sodass $|P(M_1) - P(M_2)|$ minimal ist. In der zweiten Version werden Teilmengen $\tilde{M}_1, \tilde{M}_2$ betrachtet. Dabei gilt $\tilde{M}_1 \cup \tilde{M}_2 = \{c_1, c_2, \ldots, c_n\}$ und $\tilde{M}_1 \cap \tilde{M}_2 = \emptyset$. $\tilde{M}_1$ und $\tilde{M}_2$ sollen möglichst „gleichmäßig“ gewählt werden, also $P(\tilde{M}_1) \approx P(\tilde{M}_2)$. Wir betrachten eine Variante des Algorithmus, die unter *allen* Partitionierungen stets eine „gleichmäßigste“ wählt.

a) Geben Sie eine Wahrscheinlichkeitsfunktion P an, bei der der die zweite Version Algorithmus einen Code mit größerer erwarteter Codewortlänge generiert als die erste Version des Shannon-Fano-Algorithmus.

b) *(schwieriger!)* Geben Sie ein Beispiel an, bei dem die zweite Version des Algorithmus einen Code mit kleinerer erwarteter Codewortlänge generiert als die erste Version.