Universität Mannheim
Lehrstuhl für Praktische Informatik IV
Professor Dr. W. Effelsberg
Stephan Fischer, Raum 415

Multimediasysteme
Wintersemester 1996/97
Übungsaufgaben

Multimediasysteme: Übungsblatt 1

Aufgabe 1 Huffman-Encoding Beweisen Sie folgende Aussage: Kein Baum mit den gleichen Häufigkeiten bei den äußeren Knoten hat eine kleinere gewichtete äußere Pfadlänge als der Huffman-Baum.

Lösung:

Annahme: Es gibt einen Baum, bei dem nicht die kleinsten Knotengewichte benutzt werden, um den Baum aufzubauen. Dieser hat eine kleinere gewichtete Pfadlänge als der Huffman-Baum.
Beweis per Induktion:

$n = 1$: Im Fall $n = 1$ sind beide Bäume identisch

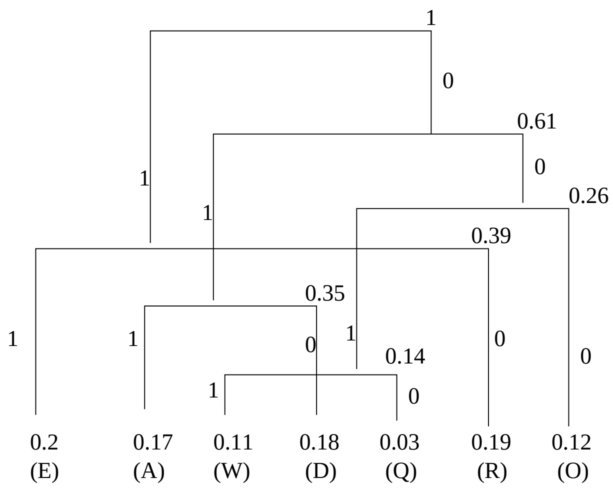
$n \rightarrow (n + 1)$: Nach Induktionsvoraussetzung wurde bis zum n-ten Schritt auf die jeweils kleinsten Gewichte geachtet. Dies erfolgt auch im jetzigen Schritt. Die gewichtete Pfadlänge war also bis zum n-ten Schritt minimal für alle gewichteten äußeren Pfade(Induktionsvoraussetzung). Da im jetzigen $(n + 1)$-ten Schritt wieder das minimale Gewicht aufaddiert wird, muß der Huffman-Baum folglich wieder minimale äußere Pfadlänge haben.

Aufgabe 2 Huffman-Encoding

Die folgende Tabelle gibt die Wahrscheinlichkeiten für einige Zeichen an. Encodieren Sie den String *EAQREDO*.

Buchstabe	Wahrscheinlichkeit
E	0.2
A	0.17
W	0.11
D	0.18
Q	0.03
R	0.19
O	0.12

Folgender Wahrscheinlichkeitsbaum kann berechnet werden:



Dies führt zu folgender Tabelle:

Buchstabe	Wahrscheinlichkeit
E	11
A	011
W	0011
D	010
Q	0010
R	10
O	000

Es ergibt sich also für $EAQREDO$:
11 011 0010 10 11 010 000. Die Ersparnis ist bei einer Länge von $\frac{8Bit}{1Zeichen}$ $1 - \frac{19}{56} =$ 0.66 entsprechend 66 Prozent.

Aufgabe 3 Ziv-Lempel Kodierung

Eine Zeichenkette bestehe aus den Buchstaben x, y und z. Welche Zeichenkette wurde komprimiert, wenn die Ziv-Lempel-Kodierung 1 2 3 4 7 3 heißt? Verifizieren Sie das Ergebnis, indem Sie die Zeichenkette wiederum mit dem Verfahren von Ziv-Lempel kodieren.

Lösung:

Decoder:

```
while((Code=GetNextCode() != EoiCode){
    if (Code == ClearCode){
        InitializeTable();
        Code = GetNextCode();
        if (Code==EoiCode)
            break;
        WriteString(StringFromCode(Code));
        OldCode = Code;
    } /* end of ClearCode case */
    else {
        if (IsInTable(Code)) {
            WriteString(StringFromCode));
            AddStringToTable(StringFromCode(OldCode)+
                FirstChar(StringFromCode(Code)));
            OldCode = Code;
        } else {
            OutString = StringFromCode(OldCode +
                FirstChar(StringFromCode(OldCode)));
            WriteString(OutString);
            AddStringToTable(OutString);
            OldCode = Code;
        }
    } }
```

Ablauf:

1. Code = 1; Output "x"; OldCode=1
2. Code = 2; Output y; xy $\to$ dict.; OldCode = 2
3. Code = 3; Output z; yz $\to$ dict.; OldCode = 3
4. Code = 4; Output xy; zx $\to$ dict.; OldCode = 4
5. Code = 7; Outstring = xyz; Output xyx; xyx $\to$ dict.; OldCode=7
6. Code = 3; Output z; xyxz $\to$ dict.; OldCode=3

Encoder:

```
InitialiseStringTable();
WriteCode(ClearCode);
ω = the empty string;
for each character in strip {
    K = GetNextChararacter();
    if ω + K is in the string table {
        ω = ω + K/* String concatenation*/
    } else {
        WriteCode(CodeFromString( ω));
        AddTableEntry( ω + K);
        ω = K
    }
} WriteCode(CodeFromString( ω));
```

Ablauf:

1. k = x; ω=x
2. k = y; Output 1; xy $\rightarrow$ dict.; ω = y
3. k = z; Output 2; yz $\rightarrow$ dict.; ω = z
4. k = x; Output 3; zx $\rightarrow$ dict.; ω = x
5. k = y; ω = xy
6. k = x; Output 4; xyx $\rightarrow$ dict.; ω = x
7. k = y; ω = xy 8. k = x; ω = xyx 9 k = z; Output 7; xyxz $\rightarrow$ dict.; ω = z