Hausaufgaben zum 23. 11. 2012

Tronje Krabbe 6435002, The-Vinh Jackie Huynh 6388888,
Arne Struck 6326505

12. Dezember 2012

1.

```
...
int b1 = (a1 << 2 | a2 >>> 4) % 256;
int b2 = (a2 << 4 | a3 >>> 2) % 256;
int b3 = (a3 << 6 | a4) % 256;
...
```

Der Modulo 256 wird benötigt, damit nur die 8 letzten Bits übernommen werden, sofern die b's nicht vorher so definiert wurden, dass sie nur die 8 letzten Bits annehmen.

2.

a)

$$\frac{360°}{15°} = 24 \text{ Codewörter}$$

b)

2 Codewörter:

0 1

4 Codewörter:

00 01
11 10

8 Codewörter:

(000) 001 011 010
110 111 101 (100)

12 Codewörter:

(die Wörter in () wurden schon hier weggelassen, um in 2 Schritten auf ein valides Ergebnis zu kommen.)

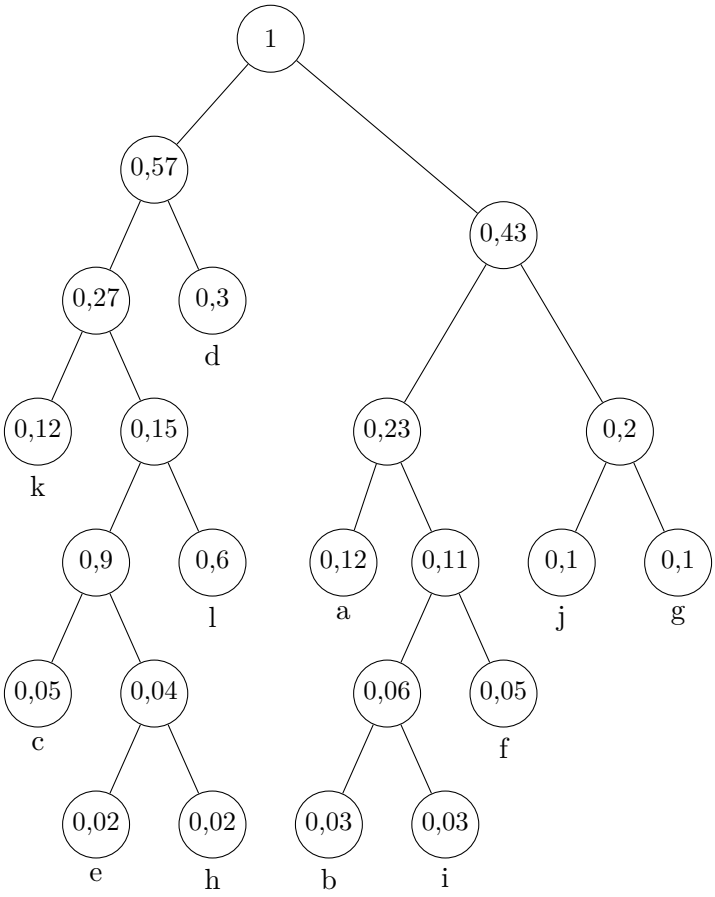
0001	0011	0010	0110	0111	0101
1101	1111	1110	1010	1011	1001

24 Codewörter:

00001	00011	00010	00110	00111	00101	01101	01111
01110	01010	01011	01001				
11001	11011	11010	11110	11111	11101	10101	10111
10110	10010	10011	10001				

3.

a)



Codierung: Linke Kante entspricht 0, rechte Kante entspricht 1.

Symbol	Codierung	Symbol	Codierung
a	100	g	111
b	10100	h	001011
c	00100	i	10101
d	01	j	110
e	001010	k	000
f	1011	l	0010

b)

$$\begin{aligned} H &= -2 \cdot 0,02 \cdot \log_2(0,02) - 2 \cdot 0,03 \cdot \log_2(0,03) - 2 \cdot 0,05 \cdot \log_2(0,05) \\ &\quad -2 \cdot 0,1 \cdot \log_2(0,1) - 2 \cdot 0,12 \cdot \log_2(0,12) - 0,3 \cdot \log_2(0,3) - 0,06 \cdot \log_2(0,06) \\ &\approx 3,125 \end{aligned}$$

4.

a)

Ziffer	0	1	2	3	4	5	6	7	8	9
Codierung	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010

$$H_0 = 10 \cdot 0,1 \cdot \log_2(2^4) = \log_2(16) = 4 \text{ Bit}$$
$$H = -10 \cdot 0,1 \cdot \log_2(0,1) \approx 3,219 \text{ Bit}$$
$$R = H_0 - H = 4 + \log_2(0,1) \approx 0,678 \text{ Bit}$$

b)

Will man die 100 verschiedenen Paare binär codieren, braucht man mindesten sieben stellige Binärworte, also eine Wortlänge von 2^7. Da aber jeder Ziffer eine Tetrade zugeordnet werden soll, muss die Wortbreite 2^8 betragen.
Eine mögliche Neucodierung wäre, analog zu a) die bisherige Codierung jeder Ziffer um eine Stelle zu verschieben und diese dann zu gruppieren.
$\Rightarrow \Big\{00010001, 00010010, ..., 10101010\Big\}$

$$H_0 = 100 \cdot 0,01 \cdot \log_2(2^8) = \log_2(128) = 8 \text{ Bit}$$
$$H = -100 \cdot 0,01 \cdot \log_2(0,01) \approx 6,643 \text{ Bit}$$
$$R_{ges} = H_0 - H = 7 + \log_2(0,01) \approx 1,356 \text{ Bit}$$
$$R = 0,01 \cdot \log_2(2^7) - 0,01 \cdot \log_2(0,01) \approx 0,1464$$

c)

Da bei 1000 verschiedenen Paaren $2^8 = 256$ offensichtlich nicht mehr ausreicht, muss die neue Wortbreite $2^{12} = 4096$ betragen.

$$H_0 = 0,001 \cdot \log_2(2^{12}) = 0,012$$

Dies gilt analog auch für 10000 Paare (4 Stellen), allerdings reicht hier auch nicht 2^{12}, da auch hier wieder zu wenig Kombinationsmöglichkeiten vorhanden sind. also ist es 2^{16}

$$H_0 = 0,0001 \cdot \log_2(2^{16}) = 0,0016$$

d)

Bei variabler Codierungslängen ergibt sich folgende Aufteilung:

0	1	2	3	4	5	6	7	8	9
0000	0001	001	010	011	100	101	110	1110	1111

$$\begin{aligned} H_0 &= \frac{1}{10} \cdot (6 \cdot 3 + 4 \cdot 4) \\ &= 3,4 \\ R &= 3,4 - \log_2(0,1) = H_0 - H \\ &\approx 0,078 \end{aligned}$$

Die Redundanz hat sich deutlich verringert.
Da sich die Redundanz aber auch mit dem Kombinieren von Zahlen zu Gruppen verringert könnte man beide Verfahren kombinieren, um die Redundanz noch weiter zu verringern.