64-041 Übung Rechnerstrukturen

Aufgabenblatt 6 Ausgabe: 23.11., Abgabe: 30.11. 24:00

Gruppe	
Name(n)	Matrikelnummer(n)

Aufgabe 6.1 (Punkte 10+10+15)

Informationstheorie: Die Dezimalziffern (0...9) werden neu codiert...

(a) Im ersten Ansatz sollen Sie eine Codierung wählen, die die Ziffern auf 4-bit Binärwörter (Nibble) abbildet. Geben Sie Ihren Code, den möglichen Informationsgehalt H_0 und die Redundanz R an.

(b) Versuchen Sie, die Redundanz zu verkleinern, indem Sie jeweils zwei Dezimalziffern zu einem Codewort zusammenfassen. Die Menge der Ausgangs-Codewörter ist deshalb $\{00, 01, 02, \ldots\ 10, 11, \ldots\ 97, 98, 99\}$.

Wie viele Bits werden für die Codewörter benötigt? Geben Sie Ihren Code und dazu den möglichen Informationsgehalt H_0 und die Redundanz R an. Wie groß ist jetzt die Redundanz bezogen auf eine einzelne Dezimalziffer?

(c) Reduzieren Sie die Redundanz, indem Sie die Dezimalziffern (0...9) auf einen Code mit variabler Länge (Fano oder Huffman) abbilden. Nehmen Sie für die Codierung gleiche Wahrscheinlichkeiten der einzelnen Ziffern an.

Geben Sie Ihren Code, den mögliche Informationsgehalt H_0 und die Redundanz R an. Wie könnte man den Code noch effizienter machen?

Aufgabe 6.2 (Punkte 5+15+5)

Optimale Codierung: Die folgenden 8 Symbole a_i sind mit ihren Wahrscheinlichkeiten $p(a_i)$ in der Tabelle angegeben:

a_i	a	b	c	d	e	f	g	h
$p(a_i)$	0,13	0,03	0,05	0,06	0,3	0,23	0,15	0,05

(a) Wie groß ist der mittlere Informationsgehalt (die Entropie) H dieser Symbole?.

(b) Bilden Sie den Huffman-Baum und geben sie die zugehörige Symbolcodierung an.

(c) Welche mittlere Codewortlänge H_0 ergibt sich?

Aufgabe 6.3 (Punkte 10+10)

2D-Paritätscode: Wir betrachten den in der Vorlesung vorgestellten zweidimensionalen Paritätscode. Jeweils 64 Datenbits werden als Matrix mit 8×8 Zeilen und Spalten notiert, und zu jeder Zeile und Spalte wird ein gerades Paritätsbit hinzugefügt. Außerdem wird noch ein weiteres Bit ganz unten rechts bestimmt, dass sich als Paritätsbit der Spalten-Paritätsbits berechnet:

$d_{0,0}$	$d_{0,1}$	$d_{0,2}$	$d_{0,3}$	$d_{0,4}$	$d_{0,5}$	$d_{0,6}$	$d_{0,7}$	$p_{0,8}$
$d_{1,0}$	$d_{1,1}$	$d_{1,2}$	$d_{1,3}$	$d_{1,4}$	$d_{1,5}$	$d_{1,6}$	$d_{1,7}$	$p_{1,8}$
$d_{2,0}$	$d_{2,1}$	$d_{2,2}$	$d_{2,3}$	$d_{2,4}$	$d_{2,5}$	$d_{2,6}$	$d_{2,7}$	$p_{2,8}$
$d_{3,0}$	$d_{3,1}$	$d_{3,2}$	$d_{3,3}$	$d_{3,4}$	$d_{3,5}$	$d_{3,6}$	$d_{3,7}$	$p_{3,8}$
$d_{4,0}$	$d_{4,1}$	$d_{4,2}$	$d_{4,3}$	$d_{4,4}$	$d_{4,5}$	$d_{4,6}$	$d_{4,7}$	$p_{4,8}$
$d_{5,0}$	$d_{5,1}$	$d_{5,2}$	$d_{5,3}$	$d_{5,4}$	$d_{5,5}$	$d_{5,6}$	$d_{5,7}$	$p_{5,8}$
$d_{6,0}$	$d_{6,1}$	$d_{6,2}$	$d_{6,3}$	$d_{6,4}$	$d_{6,5}$	$d_{6,6}$	$d_{6,7}$	$p_{6,8}$
$d_{7,0}$	$d_{7,1}$	$d_{7,2}$	$d_{7,3}$	$d_{7,4}$	$d_{7,5}$	$d_{7,6}$	$d_{7,7}$	$p_{7,8}$
$p_{8,0}$	$p_{8,1}$	$p_{8,2}$	$p_{8,3}$	$p_{8,4}$	$p_{8,5}$	$p_{8,6}$	$p_{8,7}$	$p_{8,8}$

(a) Wie groß ist die Minimaldistanz d dieses Codes? Begründen Sie Ihre Antwort.

(b) Können mit diesem Code alle Einbitfehler, Zweibitfehler, und Dreibitfehler erkannt und korrigiert werden? Warum?

Aufgabe 6.4 (Punkte 5+15)

Hamming-Code: Entsprechend dem in der Vorlesung vorgestellten Schema, wird ein 7-Bit Hamming-Code gebildet, um Einzelbitfehler korrigieren zu können. Wie in der Tabelle dargestellt, besitzt er vier Informationsbits (d_i) und drei Prüfbits (p_j). Insgesamt sind $2^4 = 16$ Informationen codierbar; die Codewörter sind in der linken Tabelle aufgelistet:

	c_1	c_2	c_3	c_4	c_5	c_6	c_7
Nr.	p_1	p_2	d_1	p_3	d_2	d_3	d_4
0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	1
2	0	1	0	1	0	1	0
3	1	0	0	0	0	1	1
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	1
6	1	1	0	0	1	1	0
7	0	0	0	1	1	1	1
8	1	1	1	0	0	0	0
9	0	0	1	1	0	0	1
10	1	0	1	1	0	1	0
11	0	1	1	0	0	1	1
12	0	1	1	1	1	0	0
13	1	0	1	0	1	0	1
14	0	0	1	0	1	1	0
15	1	1	1	1	1	1	1

Codewortstelle	c_1	c_2	c_3	c_4	c_5	c_6	c_7
Bedeutung	p_1	p_2	d_1	p_3	d_2	d_3	d_4
Prüfgruppe A	$*$		$*$		$*$		$*$
Prüfgruppe B		$*$	$*$			$*$	$*$
Prüfgruppe C				$*$	$*$	$*$	$*$

Für die Prüfstellen gilt:

$c_1 = c_3 \oplus c_5 \oplus c_7$

$c_2 = c_3 \oplus c_6 \oplus c_7$

$c_4 = c_5 \oplus c_6 \oplus c_7$

Um (einen) Einzelbitfehler zu lokalisieren, bildet man ein Prüfwort (x_a, x_b, x_c), wobei gilt:

$x_a = c_1 \oplus c_3 \oplus c_5 \oplus c_7$

$x_b = c_2 \oplus c_3 \oplus c_6 \oplus c_7$

$x_c = c_4 \oplus c_5 \oplus c_6 \oplus c_7$

Zeigen Sie anhand eines Beispiels, wie ein auftretender Einzelbitfehler lokalisiert und damit korrigiert werden kann. Angenommen Sie empfangen das Wort: $c_1 \dots c_7 = 1001010$

(a) Bilden Sie dazu die Prüfbits.

(b) Ist dieses Wort fehlerhaft? Wenn ja, wie kann man dann aus dem Prüfwort die fehlerhafte Codewortstelle bestimmen und wie sieht das korrigierte Wort aus?