64-041 Übung Rechnerstrukturen WS 15/16

Aufgabenblatt 6 Ausgabe: 18.11., Abgabe: 25.11. 24:00

Gruppe	
Name(n)	Matrikelnummer(n)

Aufgabe 6.1 (Punkte 5+10)

Codierung: Für eine Winkelcodierscheibe mit 12 ° Grad Auflösung soll ein einschrittiger zyklischer Binärcode entwickelt werden.

(a) Wie viele Codewörter hat der Code?

(b) Entwickeln Sie einen Code mit dem rekursiven Verfahren aus der Vorlesung.

Aufgabe 6.2 (Punkte 10+15+5)

Fano-Codierung: Die folgenden 10 Symbole s_i sind mit ihren Wahrscheinlichkeiten $p(s_i)$ in der Tabelle angegeben:

s_i	a	b	c	d	e	f	g	h	i	j
$p(s_i)$	0,14	0,02	0,08	0,3	0,01	0,05	0,1	0,06	0,09	0,15

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

(b) Geben Sie die Fano-Codierung für diese Symbole an.

(c) Welche mittlere Codelänge ergibt sich?

Aufgabe 6.3 (Punkte 10+10)

Huffman-Codierung: Das Huffman-Verfahren ist nicht auf Binärsymbole beschränkt. Bei der Radix-r Huffman-Codierung wird ein optimaler Code mit Codewörtern variabler Länge auf dem Alphabet der Symbole $\{0, 1, \ldots, r-1\}$ aufgebaut.

Bei der Konstruktion des Codebaums werden dabei jeweils die r Symbole (oder Knoten) mit der geringsten Symbolhäufigkeit zu einem neuen Knoten zusammengefasst. Nachdem der Baum komplett ist, werden in jeder Stufe die einzelnen Symbole $\{0, 1, \ldots, r-1\}$ zugeordnet.

Die einzige Komplikation besteht darin, dass abhängig von der Anzahl der zu codierenden Symbole eventuell ein Knoten des fertigen Codebaums weniger als r Kinder hat. Damit der erzeugte Code später minimale Länge hat, darf dieser nicht komplett „ausgelastete“ Knoten

allerdings nicht an der Wurzel des Baums sitzen (dann würden kurze Codewörter verschwendet), sondern muss ganz zu Anfang der Konstruktion (bei den längsten Codewörtern) eingefügt werden.

(a) Gegeben sei die Basis r und eine Anzahl n von zu codierenden Symbolen s_i mit Wahrscheinlichkeiten p_i, mit $i = 0, \ldots, n-1$. Überlegen Sie sich, wieviele Symbole m (mit den kleinsten Wahrscheinlichkeiten p_j) im allerersten Schritt abhängig von r und n zusammengefasst werden müssen.

(b) Das folgende Beispiel stammt aus dem Originalpaper von Huffman. Wir haben $r = 4$ und die folgenden $n = 8$ Symbole s_i mit Wahrscheinlichkeiten $p(s_i)$:

s_i	a	b	c	d	e	f	g	h
$p(s_i)$	0,22	0,20	0,18	0,15	0,10	0,08	0,05	0,02

Konstruieren Sie den Radix-4 Huffman-Codebaum für diese Symbole und geben Sie die zugehörige Huffman-Codierung an.

Aufgabe 6.4 (Punkte 5+10+10+10)

Golay(24,12,8)-Code: Wenn die Anzahl der Codewörter nicht zu groß ist, kann man fehlerkorrigierende Codes auch ohne elegante Mathematik einfach durch Ausprobieren aufbauen.

Ein Kandidat ist der Golay(24,12,8)-Binärcode, der aus insgesamt $2^{12} = 4096$ Codewörtern der Länge $n = 24$ Bit mit minimaler Hamming-Distanz $d = 8$ besteht.

Ein triviales Verfahren zur Konstruktion besteht darin, einfach alle Bitmuster der gegebenen Wortlänge (hier $n = 24$ Bit) als Kandidaten für Codewörter auszuprobieren, zum Beispiel in der Reihenfolge der Dualzahlen (also `0x000000`, `0x000001`, `0x000002`, ... `0xFFFFFF`). Ein Kandidat wird als neues Codewort akzeptiert, wenn es mindestens die geforderte Hammingdistanz (hier $d = 8$) zu allen bisher gefundenen Codewörtern aufweist.

(a) Wieviele Bitfehler in einem bei der Übertragung gestörten Codewort können mit diesem Code erkannt bzw. korrigiert werden?

(b) Verwenden Sie das oben angegebene Verfahren um ausgehend vom ersten Codewort $a_0 =$`0x000000` die nächsten drei Codewörter des Golay(24,12,8)-Codes zu berechnen.

(c) Schreiben Sie eine Java-Methode (oder C-Funktion), um die Hamming-Distanz von zwei Codewörtern (gegeben als 32-bit `int`) zu berechnen:

```
int hamming( int a, int b ) {
  ...
}
```

Erklären Sie, wie Ihr Algorithmus funktioniert.

(d) Schreiben Sie ein Java-Programm (oder C), das mit dem angegebenen Verfahren alle 4096 Codewörter des Golay(24,12,8)-Codes berechnet und ausgibt. Als Datenstruktur eignet sich zum Beispiel ein Array (der Größe 4096) oder eine `ArrayList<Integer>`.