Datenkompression (Rote), SS 2003, 1. Übungsblatt

Abgabe: Aufgabe 1 und 2: bis zum Donnerstag, 24.4.2003, in der Vorlesung
Aufgabe 3: bis zum Montag, 28.4.2003, im Sekretariat (Raum 111)

1. (10 Punkte) Wie viele verschiedene Nachrichten der Länge n lassen sich aus den Codewörtern aba, $aaaab$, bb, ba bilden? Jedes Codewort darf dabei beliebig oft verwendet werden. Geben Sie eine Rekursionsformel für die Anzahl A_n der Nachrichten der Länge n an.

2. (10 Punkte) Man kann Dateien mit einer Länge von vier Bit auf folgende Weise komprimieren. Wenn die Datei nur aus Nullen besteht, wird sie unverändert abgespeichert. Andernfalls wird die erste Eins, zusammen mit allen eventuell davor stehenden Nullen, entfernt und nur der Rest abgespeichert.

   Auf diese Weise werden fast alle 4-Bit-Dateien (mit einer einzigen Ausnahme) um mindestens 1 Bit verkürzt.

   (a) (5 Punkte) Zeigen Sie, dass man aus den komprimierten Dateien die ursprünglichen Dateien eindeutig wiederherstellen kann.
   (b) (5 Punkte) Berechnen Sie die durchschnittliche Länge der entstehenden komprimierten Dateien unter der Annahme, dass alle 4-Bit Dateien gleich wahrscheinlich sind.
   (c) Diskutieren Sie, ob diese Methode geeignet verallgemeinert werden kann, sodass es für ein praktisches Kompressionsverfahren in Frage kommt.

3. (10 Punkte) Welche der folgenden Codes sind präfixfrei? Welche sind eindeutig entzifferbar?

   (a) (2 Punkte) 001, 0100, 110, 10, 111.
   (b) (2 Punkte) 001, 0100, 110, 010, 111, 01.
   (c) (6 Punkte) b, ba, ba^7, ba^{13}, ba^{14}; a^3b, a^3ba^2, a^3ba^4, a^3ba^6; a^8b, a^8ba^2, a^8ba^4, a^8ba^6; $a^{11}b$, $a^{11}ba$, $a^{11}ba^2$. (a^{14} bedeutet 14 Wiederholungen des Buchstabens a, usw.)

4. (0 Punkte) Gegeben sei eine Alphabet $S = \{a_1, \ldots, a_n\}$ mit n Symbolen und eine Wahrscheinlichkeitsverteilung $p = (p_1, \ldots, p_n)$ mit Entropie $H(p)$. Rechen Sie nach, dass das Alphabet S^2, in dem das „Symbolpaar“ a_ia_j mit Wahrscheinlichkeit p_ip_j vorkommt, Entropie $2H(p)$ hat.

5. (0 Punkte) Finden Sie den Fehler im folgenden „Beweis“, dass die Kraft'sche Ungleichung hinreichend für die Existenz eines präfixfreien Codes ist.

   > Gegeben seien die Längen $l_1, l_2, \ldots, l_n$, die die Kraft'sche Ungleichung erfüllen. Die maximale Länge sei $h = l_{\max}$. Wir beginnen mit einem vollständigen binären Baum der Höhe h mit 2^h Blättern. Jeder Knoten des Baumes auf Tiefe t entspricht einem 0-1-Wort der Länge t. Für das erste Codewort wählen wir einen Knoten der Tiefe l_1. Dadurch wird der Teilbaum, der unter diesem Knoten hängt, für zukünftige Codewörter gesperrt; es werden somit 2^{h-l_1} Blätter eliminiert. Als nächstes wird ein verbleibender Knoten auf Tiefe l_2 ausgewählt und dadurch 2^{h-l_2} Blätter eliminiert, und so weiter. Man kann dabei niemals stecken bleiben, denn aus der Kraft'schen Ungleichung ergibt sich, dass bei der Wahl des k-ten Wortes noch mindestens
   >
   > $$2^h - (2^{h-l_1} + 2^{h-l_2} + \cdots + 2^{h-l_{k-1}}) \geq 2^{h-l_k}$$
   >
   > Blätter übrig sind.

Datenkompression (Rote), SS 2003, 2. Übungsblatt
Abgabe: bis Freitag, 2.5.2003, in der Vorlesung

6. Kapazität eines Kanals (5 Punkte)

   Das Morse-Alphabet ist ein Kode über dem Alphabet $\{\cdot, -, \sqcup\}$. Nehmen wir an, dass ein Punkt („$\cdot$") und ein Zwischenraum („$\sqcup$") je 0,2 sec dauert und ein Strich („$-$") 0,4 sec.

   Wie viele verschiedene Nachrichten lassen sich in $n \cdot 0{,}2$ Sekunden übertragen? (Geben Sie eine Rekursionsformel an.) Wie groß ist die Kapazität dieses Kanals (in bit/sec)?

7. (0 Punkte) Finden Sie einen eindeutig entzifferbaren Kode, bei dem es beliebig lange Nachrichten gibt, die man erst dekodieren kann, wenn man das letzte Bit gelesen hat.

8. (0 Punkte) Verallgemeinern Sie den Huffman-Algorithmus, sodass er einen optimalen Kode für ein Zielalphabet mit mehr als zwei Symbolen erzeugt.

9. (0 Punkte)

   (a) Beweisen Sie die folgende Beziehung für die Entropie:

   $$H(p_1\alpha, p_1(1-\alpha), p_2, \ldots, p_n) = H(p_1, p_2, \ldots, p_n) + p_1 \cdot H(\alpha, 1-\alpha),$$

   for $0 \le \alpha \le 1$.

   (b) Beweisen Sie die folgende Ungleichung für die Entropie:

   $$0 \le H(p_1, p_2, \ldots, p_n) \le H(\tfrac{1}{n}, \tfrac{1}{n}, \ldots, \tfrac{1}{n}) = n \log_2 n$$

   Für welche Verteilungen $(p_1, p_2, \ldots, p_n)$ wird in der linken Ungleichung Gleichheit erzielt?

10. Huffman-Nummerierung (15 Punkte)

    Wir nummerieren die Knoten des Kodebaumes, der beim binären Huffman-Algorithmus entsteht, in der Reihenfolge, in der die Knoten gepaart werden. Das heißt, die beiden Blätter mit den kleinsten Gewichten bekommen die Nummern 1 und 2, und die Wurzel hat Nummer $2n-1$. Das *Gewicht* eines Blattes ist seine Wahrscheinlichkeit, und das Gewicht eines inneren Knotens ist die Summe der Gewichte aller Blätter.

    Beweisen Sie, dass diese Nummerierung folgende Eigenschaften hat.

    (a) Knoten mit größerem Gewicht haben größere Nummern.

    (b) Zwei Geschwisterknoten haben aufeinanderfolgende Nummern.

    (c) Alle Knoten auf Tiefe i haben größere Nummern als die Knoten auf Tiefe $i+1$.

    (d) Betrachte zwei Knoten u und v auf Tiefe i. Wenn u eine kleinere Nummer als v hat, dann haben die Kinder von u eine kleinere Nummer als die Kinder von v (sofern diese Kinder vorhanden sind).

    Beweisen Sie, dass ein Baum, dessen Gewichte in den Blättern die gegebenen Wahrscheinlichkeiten p_i sind, und für den eine Nummerierung mit den obigen Eigenschaften existiert, ein Huffman-Baum ist.

11. (10 Punkte) Überlegen Sie, wie man einen optimalen binären Kode möglichst platzsparend abspeichern kann.

    Wieviele Bits benötigen Sie für eine Kode für das Alphabet Σ der 10 Ziffern? Wieviele Bits benötigen Sie, um alle 256 möglichen Bytes zu kodieren?

Datenkompression (Rote), SS 2003, 3. Übungsblatt

Abgabe: bis Freitag, 9.5.2003, in der Vorlesung; Programmieraufgabe bis 23.5.2003.

12. (0 Punkte) Die Länge eines Kodeworts für ein Symbol mit Wahrscheinlichkeit p_i bein Shannon-Kode ist $\lceil -\log_2 p_i \rceil$. Kann es sein, dass dieses Symbol bei einem optimalen Kode eine Länge größer als $\lceil -\log_2 p_i \rceil$ hat?

13. Der Golomb-Kode (10 Punkte)

    Bestimmen Sie einen optimalen Kode für die *geometrische Verteilung*

    $$p_i = p \cdot (1-p)^i, \text{ für } i = 0, 1, 2, \ldots$$

    für den Parameter (a) $p = 0{,}6$ (b) $p = 0{,}5$ (c) $p = 0{,}125$. (p_i ist die Wahrscheinlichkeit, dass man bei der Wiederholung eines Experiments, das Erfolgswahrscheinlichkeit p hat, i Misserfolge hat, bevor das Experiment das erste Mal gelingt.)

    (Achtung: Da es unendlich viele Blätter gibt, kann man den Huffman-Algorithmus nicht ohne Weiteres anwenden.)

    Zusatzfrage zu Aufgabe (c): Wenn man eine zufällige 0-1-Folge in Dreierblöcken liest, dann ist p_i die Wahrscheinlichkeit, dass man i Blöcke lesen muss, bevor man einen Block 000 trifft. Man kann dementsprechend einen Kode mit folgender Dekodierungsregel verwenden:

    > Lies die kodierte Nachricht in Dreierblöcken bis zu einem 000-Block. Das dekodierte Symbol i ist die Anzahl der gelesenen Blöcke minus 1.

    Warum ist der Kode, der durch diese Regel dekodiert wird, kein (guter) Kode für die Aufgabe?

14. (0 Punkte) Unter welchen Bedingungen an die Wahrscheinlichkeiten $p_1, \ldots, p_n$ haben alle Kodewörter im Huffman-Kode die gleiche Länge?

15. (5 Zusatzpunkte) Beim Huffman-Algorithmus kann es vorkommen, dass mehrere Knoten das gleiche Gewicht haben und die Auswahl des nächsten Schrittes nicht eindeutig ist. Wie kann man in diesem Fall die Auswahl der beiden nächsten Knoten zum Vereinigen steuern, sodass man möglichst keine langen Kodewörter erhält? (Genau gesagt: damit das längste Kodewort möglichst kurz ist.)

16. (0 Punkte) Versuchen Sie eine Verteilung zu finden, bei der das längste Kodewort in einem optimalen Kode mindestens Länge L hat und die kleinste Wahrscheinlichkeit möglichst groß ist, für verschiedene Werte von L.

17. (25 Punkte, Abgabe bis Freitag 23. Mai, in der Vorlesung) Schreiben Sie ein Programm, das eine Datei einliest und als Folge von Bytes mit dem Huffman-Algorithmus komprimiert. Wählen Sie eine der beiden folgenden Möglichkeiten:

    (a) Die Häufigkeiten werden in einem ersten Durchlauf bestimmt; der Huffman-Kode wird am Anfang der komprimierten Datei gespeichert. (Sie können dabei auf Aufgabe 11 zurückgreifen.)

    (b) Verwenden Sie einen adaptiven Huffman-Code.

Datenkompression (Rote), SS 2003, 4. Übungsblatt
Abgabe: bis Freitag, 16. 5. 2003, in der Vorlesung

18. Optimale Kodes als ganzahliges lineares Optimierungsproblem (20 Punkte)

    (a) (5 Punkte) B_l und I_l sei die Anzahl der Blätter und der inneren Knoten auf Tiefe l in einem binären Kode-Baum. Zeigen Sie, dass diese Zahlen durch folgende Gleichungen charakterisiert sind.

    $$B_0 + I_0 = 1, \text{ und } B_l + I_l = 2I_{l-1} \text{ für } l = 1, 2, \ldots$$

    (b) (5 Punkte) R_l ist die Anzahl der Blätter auf Tiefe l und tiefer:

    $$R_0 = n, \quad R_l = R_{l-1} - B_{l-1} \text{ für } l = 1, 2, \ldots$$

    Das Gewicht (die mittlere gewichtete Pfadlänge) eines optimalen Baumes mit den Parametern n, B_l und I_l für die Gewichte $p_1 \leq p_2 \leq \cdots \leq p_n$ ist durch

    $$\sum_{l=0}^{\infty} f(R_l)$$

    gegeben, wobei $f(i) = p_1 + p_2 + \cdots + p_i$ ist.

    (c) (5 Punkte) Das Gewicht kann man auch durch folgende Zielfunktion ausdrücken:

    $$\text{minimiere } \sum_{l=0}^{\infty} y_l$$

    unter den Bedingungen

    $$y_l \geq p_1 + p_2 + \cdots + p_i + p_{i+1}(R_l - i), \text{ für } i = 0, 1, \ldots, n-1 \text{ und } l \geq 0$$

    Hinweis: Zeichnen Sie ein Diagramm für die Beziehung zwischen y_l und R_l.

    (d) (5 Punkte) Manchmal möchte man einen *längenbeschränkten* optimalen Kode, einen Kode, bei dem alle Kodewörte durch eine vorgegebene Länge L beschränkt sind. Durch welche zusätzlichen Bedingungen lässt sich das obige Modell für diesen Fall anpassen?

19. Optimale Kodes mit zusätzlicher Nebenzielfunktion (0 Punkte)
    Manchmal gibt es mehrere optimale Kodes, die die Zielfunktion $\sum_{i=1}^{n} p_i l_i$ minimieren. Dann sucht man unter den optimalen Kodes einen, der möglichst keine allzu langen Kodewörter enthält. (Siehe auch Aufgabe 14.) Eine Möglichkeit ist, unter den optimalen Kodes jenen auszuwählen, der einfach die *ungewichtete* mittlere Länge, d.h, die Zielfunktion $\sum_{i=1}^{n} l_i$ minimiert. Dieses Problem kann man durch die Zielfunktion

    $$\text{minimiere } \sum_{i=1}^{n} p_i l_i + \varepsilon \cdot \left(\sum_{i=1}^{n} l_i \right)$$

    ausdrücken, für ein genügend kleines $\varepsilon > 0$.

    (a) Wie klein muss man ε wählen, damit dieser Ansatz funktioniert?

    (b) Wie kann man dieses Problem durch einen geeignete Variante des Huffman-Algorithmus lösen?

    (c) (Zusatzfrage, 10 Punkte) Wird durch die Optimallösung automatisch auch das *längste* Kodewort minimiert? (Geben Sie einen Beweis oder ein Gegenbeispiel.)

Datenkompression (Rote), SS 2003, 5. Übungsblatt
Abgabe: bis Freitag, 23. 5. 2003, in der Vorlesung

20. (15 Punkte) Kodierung einer geordneten Auswahl ohne Wiederholung

    Man möchte eine Folge von k verschiedenen Symbolen aus einem vorgegebenen Grundalphabet mit n Symbolen kodieren (zum Beispiel die Blätter eines Huffman-Baumes). Nehmen wir an, dass k und n bekannt sind und nicht mehr kodiert werden müssen.

    (a) (5 Punkte) Überlegen Sie sich einen Algorithmus zur platzsparenden Darstellung dieser Information. Geben Sie näherungsweise die benötigte Anzahl von Bits Ihres Verfahrens an, für große Werte von n, k, und $n-k$.

    (b) (5 Punkte) Aus dem binären Logarithmus für die Anzahl der Möglichkeiten ergibt sich die *informationstheoretische untere Schranke I*: Jedes Kodierungsverfahren für diese Aufgabe benötigt im schlimmsten Fall mindestens I Bits. Wie groß ist die informationstheoretische untere Schranke für große Werte von n, k, und $n-k$, angenähert? (Hinweis: Verwenden Sie die Stirling'sche Formel $n! \approx \left(\frac{n}{e}\right)^n \cdot \sqrt{2\pi n}$.)

    (c) (5 Punkte) Bestimmen Sie die genaue benötigte Anzahl von Bits für (a) und (b) für die Fälle $n = 256$ und $k = 100$, sowie für $n = 10000$ und $k = 50$.

    (d) (0 Punkte) Durch welches Verfahren kann man (im Prinzip) die informationstheoretische untere Schranke exakt erreichen?

21. (15 Punkte) Versuchen Sie, einen längenbeschränkten Kode für die Gewichte p_i zu erzeugen, indem Sie die Gewichte p_i durch $p'_i := \max\{p_i, s\}$ für einen geeignet gewählten Parameter s ersetzen und auf die neuen Gewichte den Huffman-Algorithmus anwenden. Der Parameter s soll durch Probieren ermittelt werden.

    (a) (0 Punkte) $(p_1, \ldots, p_n) = (1, 1, 3, 5, 6, 11, 13)$ und die Längenschranke $L = 3$.

    (b) (5 Punkte) $(p_i) = (1, 1, 3, 5, 6, 11, 13)$ und $L = 4$.

    (c) (5 Punkte) $(p_i) = (1, 3, 4, 7, 11, 18, 29, 47, 76)$ und $L = 5$.

    (d) (0 Punkte) $(p_i) = (1, 1, 2, 3, 5, 8, 13, 21, 34)$ und $L = 4$.

    (e) (0 Punkte) $(p_i) = (1, 1, 1, 2, 3, 15, 18, 130, 200)$ und $L = 4$.

    (f) (5 Punkte) Wie ändert sich die Länge des längsten Kodewortes, wenn man s vergrößert/verkleinert? (Beweisen Sie Ihre Antwort.)

    (g) (0 Punkte) Wenn es mehrere verschiedene Werte von s gibt, die zur gleichen maximalen Kodewortlänge führen, ist es dann günstiger, ein größeres oder eine kleineres s zu wählen? (Diese Frage können Sie auch experimentell untersuchen.)

    (h) Zusatzfrage (15 Punkte). Kann man durch geignete Wahl von s auf diese Weise immer einen *optimalen* längenbeschränkten Kode erzeugen? Finden Sie einen Beweis oder ein Gegenbeispiel.

    (i) Zusatzfrage (5 Punkte). Kann es passieren, dass man durch keine Wahl von s einen Kode bekommt, bei dem die größte Kodewortlänge genau gleich L ist? Finden Sie einen Beweis oder ein Beispiel.

22. (0 Punkte) Gibt es zu dem Kode aus Aufgabe 3c einen *präfixfreien* Kode über dem Alphabet $\{a, b\}$, bei dem jedes der 16 Quellsymbole durch ein Kodeword mit genau gleich vielen a's und b's dargestellt wird wie im Kode aus Aufgabe 3c: b, ba, ba^7, ba^{13}, ba^{14}; a^3b, a^3ba^2, a^3ba^4, a^3ba^6; a^8b, a^8ba^2, a^8ba^4, a^8ba^6; $a^{11}b$, $a^{11}ba$, $a^{11}ba^2$.

Datenkompression (Rote), SS 2003, 6. Übungsblatt

Abgabe: Aufgabe 26 bis Montag, 2. 6. 2003, der Rest bis Montag, 9. 6. 2003.

23. (5 Punkte) Beweisen Sie folgenden Hilfssatz: Gegeben ist eine Folge von n Zahlen a_i der Form 2^{-l} mit $l \geq 0$, mit Summe A. Für jede natürliche Zahl $b \leq A$ gibt es eine Teilmenge $I \subseteq \{1, \ldots, n\}$ mit Summe $\sum_{i \in I} a_i = b$.

24. (10 Punkte) Vergleichen Sie die Kompressionsrate des adaptiven Huffman-Kodes und eines statischen Huffman-Kodes für folgende Eingaben:

    (a) $a \ldots ab \ldots bc \ldots cd \ldots d$
    (b) $a \ldots ab \ldots bc \ldots cd \ldots de$
    (c) $a \ldots ab \ldots bc \ldots c \cdots g \ldots gh \ldots hi$

    Es handelt sich dabei um vier beziehungsweise acht gleich lange Blöcke von gleichen Buchstaben, denen im Fall (b) und (c) ein einzelnes Symbol folgt.

    Bestimmen Sie die durchschnittlich benötigte Zahl von Bits pro Eingabezeichen, wenn die Länge der Nachricht gegen unendlich geht.

    Zusatzfrage (0 Punkte). Versuchen Sie, große Beispiele zu finden, bei dem das Längenverhältnis zwischen adaptiver und statischer Huffman-Kodierung möglichst groß und möglichst klein ist. Suchen Sie auch Beispiele wo der Unterschied zwischen adaptiver und statischer Huffman-Kodierung bezogen auf die mittlere Anzahl der Bits pro Eingabesymbol möglichst groß ist, und zwar in beiden Richtungen.

25. (0 Punkte) Beweisen Sie: In einem Baum, der vom Huffman-Algorithmus erzeugt worden ist, haben die vier Kinder von zwei inneren Knoten mit gleichem Gewicht ebenfalls alle das gleiche Gewicht. (Stimmt diese Aussage auch, wenn es Knoten vom Gewicht 0 gibt?)

26. Experimente mit Huffman-Kompression (Programmieraufgabe, 10 Punkte)
    Wenden Sie Ihr Programm aus Aufgabe 17 auf die sechs Dateien `abracadabra`, `book1`, `4letter.words`, `hw2p3b.wav`, `paper1`, und `progc` in

    `http://page.inf.fu-berlin.de/~rote/Lere/2003-SS/Datenkompression/Daten/`

    an. Bestimmen Sie die Länge jeder entstehenden komprimierten Datei in Bits insgesamt und in Bits pro Symbol der Eingabedatei. Schlüsseln Sie die entstehenden Gesamtlängen folgendermaßen auf:

    (a) Die Bits für den Huffman-Kode (sofern Sie keinen adaptiven Kode gewählt haben.)
    (b) Gegebenenfalls die Bits für die Übertragung des Alphabets (entweder als Blätter der Kode-Baumes oder als "neue" Symbole beim adaptiven Kodieren)
    (c) Die Bits für kodierte Nachricht, die aus den Wörter des (adaptiven) Huffman-Kodes selbst besteht.
    (d) Eventuelle zuzätzliche Bits, zum Beispiel zur Speicherung der Dateilänge oder zum Auffüllen auf volle Bytes.

27. (5 Punkte) Runden bei der arithmetischen Kodierung.
    Nehmen wir an, bei der arithmetischen Kodierung arbeiten wir mit ganzzahligen Intervallgrenzen. Wir haben zwei mögliche Symbole mit Wahrscheinlichkeiten $p_0 = 0{,}5555$ und $p_1 = 0{,}4445$ zu kodieren. Wie sollen wir in diesem Fall ein Intervall der Länge $L = 1000$ in zwei ganzzahlige Intervalle zerlegen, sodass die mittlere erwartete Nachrichtenlänge möglichst klein wird, wenn wir annehmen, dass p_0 und p_1 die tatsächlichen Wahrscheinlichkeiten für das Auftreten der Symbole sind.

Datenkompression (Rote), SS 2003, 7. Übungsblatt
Abgabe: bis Montag, 16. 6. 2003

28. Verzögerung bei der arithmetischen Kodierung. (10 Punkte)
Wieviele Eingabesymbole muss man bei der arithmetischen Kodierung im schlimmsten Fall lesen, bevor man das erste Bit der kodierten Nachricht ausgeben kann? Wieviele Bits muss man bei der Dekodierung im schlimmsten Fall lesen, bevor man das erste Zeichen der Nachricht ausgeben kann?

    - (5 Punkte) Nehmen Sie an, dass mit exakter Arithmetik gerechnet wird.
    - (5 Punkte) Nehmen Sie an, dass die Intervalllängen auf Vielfache von 1/100 gerundet werden.
    - (0 Punkte) Nehmen Sie an, dass die Intervalllängen auf Vielfache von $1/N$ gerundet werden. (Das heißt mit anderen Worten: Es wird ganzzahlige Arithmetik im Bereich $[0, N]$ verwendet.)

    Beweisen Sie die oberen Schranken, und geben Sie jeweils auch Beispiele an, wo die von Ihnen behaupteten Verzögerungen auftreten.

29. Sequentielle Suche bei der arithmetischen Dekodierung. (11 Punkte)
Bei der Dekodierung muss man das Teilintervall finden, das die Kodezahl enthält. Eine einfache Möglichkeit besteht darin, die Intervalle sequentiell zu durchsuchen. Dabei ist es natürlich vorteilhaft, die Intervalle der Länge nach absteigend zu sortieren. Formal sieht das so aus: Gegeben ist eine Folge von Häufigkeiten $m_1 \geq m_2 \geq \cdots \geq m_n$ mit Summe $M = m_1 + \cdots + m_n$. Für eine zufällige Zahl x im Intervall $[L, U)$ soll der Index i bestimmt werden, sodass
$$L + (U - L)\frac{m_1 + \cdots + m_{i-1}}{M} \leq x < L + (U - L)\frac{m_1 + \cdots + m_i}{M}$$
ist, und zwar durch sequentielles Durchprobieren der Werte $i = 1, 2, \ldots, n$.

    - (0 Punkte) Wie viel Zeit benötigt dieses Verfahren, wenn x im i-ten Intervall liegt?
    - (2 Punkte) Schreiben Sie eine Formel für die erwartete Suchzeit.
    - (2 Punkte) Berechnen Sie die erwartete Suchzeit im Fall, dass alle Symbole gleich häufig sind.
    - (2 Punkte) Zeigen Sie, dass dieser Fall der schlechteste mögliche Fall ist.
    - (3 Punkte) Berechnen Sie die erwartete Suchzeit für den Fall, wo m_i annähernd proportional zu $1/i$ ist. (Diese Verteilung entspricht dem sogenannten Zipf'schen Gesetz.)
    - (2 Punkte) Berechnen Sie die erwartete Suchzeit für den Fall, wo m_i annähernd proportional zu $1/i^2$ ist.

30. Programmieraufgabe (10 Punkte)
Schreiben Sie ein Programm zur Dekodierung der mit Ihrem Programm aus Aufgabe 17 komprimierten Daten, und wenden Sie es auf die Beispiele aus Aufgabe 26 an. (Wenn Sie Ihr Kodierungsprogramm ändern müssen, gibt es 5 Punkte Abzug.)

31. Erweitern nicht-vollständiger Präfix-Kodes (0 Punkte)
Ein Kode heißt *vollständig*, wenn jede Kette $x \in T^*$ über dem Kodealphabet T das Anfangsstück einer kodierten Nachricht ist. Beweisen Sie, dass man jeden Präfix-Kode durch Hinzunahme zusätzlicher Kodewörter zu einem vollständigen Präfix-Kode ergänzen kann.

Datenkompression (Rote), SS 2003, 8. Übungsblatt

Abgabe: bis Montag, 30. 6. 2003

32. Kompression nach LZW (Ziv-Lempel und Welch) (8 Punkte)

    (a) Dekodieren Sie die Folge

    1 2 4 3 1 4 9 5 8 12 2

    mit einem Wörterbuch, das am Anfang die Buchstaben a, b, c enthält.

    (b) Kodieren Sie die Nachricht *ababababacccdccdccdca* über dem Alphabet a, b, c, d.

33. (7 Punkte) Zeigen Sie, dass für die N-te Einheitswurzel $\omega = e^{2\pi i/N} = \cos\frac{2\pi}{N} + i\sin\frac{2\pi}{N}$ und $k \in \mathbb{Z}$ folgende Beziehung gilt:

$$\sum_{j=0}^{N-1} \omega^{jk} = \begin{cases} N, & \text{falls } k \text{ ein Vielfaches von } N \text{ ist,} \\ 0, & \text{sonst.} \end{cases}$$

    (Hinweis: Die Summe ist eine geometrische Reihe.)

34. Fouriertransformation (15 Punkte, wird als Programmieraufgabe bewertet)

    (a) Berechnen Sie die diskrete Fouriertransformation für folgende Vektoren

        i. $(2, 3, 4, 5, 6, 7, 8, 9)$
        ii. $(5, 2, -1, -2, -2, 0, 2, 4)$
        iii. $(2, 0, 2, 0, 2, 0, 2, 0)$

    (b) Setzen sie alle Fourierkoeffizienten bis auf die k Fourierkoeffizienten mit dem größten Betrag auf 0, und berechnen Sie vom Ergebnis die inverse Fouriertransformation, für $k = 1, 2, 3$.

    (c) Vergleichen Sie das Resultat mit den ursprünglichen Daten. Für welche Werte von k bekommt man eine gute Annäherung?

35. (0 Punkte) Redundanz eines Kodes.

    Die *bedingte Entropie* h_k in einem inneren Knoten k eines vollständigen Kodebaumes mit Gewicht p und Kindern mit Gewichten p_1 und p_2 ($p_1 + p_2 = p$) ist $H(p_1/p, p_2/p)$. Beweisen Sie die folgenden Aussagen durch Induktion.

    Die Entropie ist

$$H(p_1, \ldots, p_k) = \sum_k p_k h_k,$$

    wobei die Summation über alle inneren Knoten k des Baumes läuft.

    Das Gewicht (die gewichtete mittlere Kodelänge) ist

$$E[l_i] = \sum_{i=1}^{n} p_i l_i = \sum_k p_k h_k,$$

    wobei die Summation über alle inneren Knoten k des Baumes läuft.

    Die *lokale Redundanz* in einem inneren Knoten k ist $r_k := p_k(1 - h_k)$.

$$E[l_i] - H(p_1, \ldots, p_k) = \sum_k r_k,$$

    Unter welchen Bedingungen ist dieser Ausdruck null?

Datenkompression (Rote), SS 2003, 9. Übungsblatt
Abgabe: bis Montag, 7. 7. 2003

36. (7 Punkte) Leiten Sie die Formel für die inverse diskrete Cosinustransformation her:

$$g_j = \frac{1}{\sqrt{N}} G_0 + \sqrt{\frac{2}{N}} \sum_{k=1}^{N-1} G_k \cos \frac{\pi k(2j+1)}{2N}$$

(Hinweis: Sie können zum Beispiel die Beziehung zur diskreten Fouriertransformation ausnützen und die Herleitung in der Vorlesung rückwärts verfolgen.)

37. (8 Punkte) Bei der Bildkompression mit JPEG sind die Eingabewerte üblicherweise ganze Zahlen im Bereich $0 \ldots 255$. Vor der diskreten Cosinustransformation wird eine *Verschiebung des Nullpunkts* durchgeführt: Von allen Werten wird 128 subtrahiert, sodass die Werte g_{ij}, die der diskreten Cosinustransformation unterzogen werden, im Bereich $-128 \ldots 127$ liegen. Untersuchen Sie, wie sich diese Verschiebung auf die transformierten Werte G_{mn} $(0 \leq m, n \leq 7)$ auswirkt.

    (Zusatzfrage: Welchen Vorteil bringt die Verschiebung des Nullpunktes?)

38. Diskrete Cosinustransformation (15 Punkte, wird als Programmieraufgabe bewertet)

    (a) Berechnen Sie die diskrete Cosinustransformation für folgende Vektoren

        i. $(2, 3, 4, 5, 6, 7, 8, 9)$
        ii. $(5, 2, -1, -2, -2, 0, 2, 4)$
        iii. $(2, 0, 2, 0, 2, 0, 2, 0)$. (Das sind dieselben Vektoren wie bei Aufgabe 34.)
        iv. $(a_0, a_1, \ldots, a_7)$ mit $a_i = \sin(1.178\, i + 2.16)$. (Winkel im Bogenmaß)
        v. $(a_0, a_1, \ldots, a_7)$ mit $a_i = \cos(1.178\, i)$.
        vi. $(a_0, a_1, \ldots, a_7)$ mit $a_i = \cos(1.278\, i)$.

    (b) Setzen sie alle transformierten Koeffizienten bis auf die k Koeffizienten mit dem größten Betrag auf 0, und berechnen Sie vom Ergebnis die inverse Kosinustransformation, für $k = 1, 2, 3$.

    (c) Vergleichen Sie das Resultat mit den ursprünglichen Daten. Für welche Werte von k bekommt man eine gute Annäherung?

        Vergleichen Sie die Ergebnisse auch mit den Ergebnissen von Aufgabe 34 (Fouriertransformation).

Datenkompression (Rote), SS 2003, 10. Übungsblatt
Abgabe: bis Montag, 14. 7. 2003

39. (0 Punkte) Aus 4 ganzzahligen Werten a, b, c, d wird der gerundete Mittelwert $\bar{x} = \lfloor \frac{a+b+c+d}{4} + 1/2 \rfloor$ gebildet. Kann man aus $\bar{x}$ und a, b, c den Wert d rekonstruieren? Wie viele Bits benötigt man zusätzlich, um d eindeutig zu bestimmen?

40. (a) (7 Punkte) Aus 4 ganzzahligen Werten a, b, c, d wird der gerundete Mittelwert $\bar{x} = \lfloor \frac{a+b+c+d}{4} + 1/2 \rfloor$ gebildet. Wie kann man aus $\bar{x}$ und den Differenzen $\Delta_1 = a - b$, $\Delta_2 = a - c$, $\Delta_3 = d - c$ die Werte a, b, c, d rekonstruieren?

    (b) (7 Punkte) Bei der hierarchischen Darstellung werden jeweils 4 Bildpunkte in einer Ebene der Hierarchie zu einem Bildpunkt der höheren Ebene zusammengefasst.

    Wenn man von einem 1024×1024-Gitter mit 8-Bit-Grauwerten zwischen 0 und 255 ausgeht und das Verfahren aus Teil (a) anwendet, wieviele Bits muss man dann (ohne Kompression) im schlimmsten Fall zur Übertragung aller 11 Hierarchiestufen (beginnend mit der höchsten Hierarchie mit nur einem einzigen Punkt) vorsehen?

    (c) (0 Punkte) Wie ist es, wenn man statt dessen das Verfahren aus Aufgabe 39 anwendet? Gibt es ein Verfahren für die hierarchische Darstellung, das sparsamer ist? (Hinweis: Beachten Sie Aufgabe 20.)

41. (8 Punkte) Es gibt auch einen JPEG-Standard für verlustfreie Bildkompression[1] (JPEG-LS). Der Kern der Verfahrens ist die Vorhersage eines Pixelwertes aus den drei benachbarten Bildpunkten in Richtung N (Nord), NW (Nordwest) und W (West). Und zwar wird als Vorhersagewert der Median (der mittlere Wert) der drei Werte N, W und $N - NW + W$ gewählt.

    Betrachten sie ein 10×10 Pixelarray (a_{ij}) $(0 \leq i, j \leq 9)$ mit folgenden Werten:

    (a) $a_{ij} = 10 + 3i - j$
    (b) $a_{ij} = 3i + 2j$
    (c) Das Bild ist durch eine horizontale Linie in zwei Hälften geteilt; in jeder Hälfte ist der Wert konstant.
    (d) Das Bild ist durch eine vertikale Linie in zwei Hälften geteilt; in jeder Hälfte ist der Wert konstant.

    Nehmen Sie an, dass die erste Zeile ($i = 0$) und die erste Spalte ($j = 0$) direkt übertragen wird, und bestimmen Sie den vorhergesagten Wert für die übrigen Punkte, sowie den Fehler zum tatsächlichen Wert.

    (Zusatzfrage: Wie kann man die oben beschriebene Bestimmung des Vorhersagewertes möglichst einfach programmieren?)

[1] http://www.hpl.hp.com/loco/