Praktische Informatik II SS 2007

Übungsblatt 8

Abgabe: 25.06.2007

Aufgabe 1 Huffman - Baum

Das Ziel der Aufgabe ist es, eine Zeichenkodierung mit Hilfe eines Baumes zu implementieren. Es sollen Zeichen, die häufig vorkommen, mit einer möglichst kurzen Bitfolge kodiert werden. Dadurch werden Zeichen, die selten vorkommen, mit einer eventuell längeren Bitfolge kodiert. Bei einer guten Kodierung wird dadurch eine Datenkompression erreicht. Die Berechnung der Kodierungsbitfolge für jedes Zeichen wird mit Hilfe eines Huffman Baums durchgeführt.

Ein Huffman Baum ist ein Binärbaum, der auf den Kanten die Markierung 0 oder 1 besitzt. In den Knoten des Baumes sind Wahrscheinlichkeiten gespeichert, in den Blattknoten zusätzlich die zu kodierenden Zeichen. Ein Pfad von der Wurzel zu einem Blatt ergibt die Bitfolge, die dem Huffman Code des Zeichens im Blatt entspricht. Dazu werden die Markierungen auf den Kanten konkateniert. Die linke Nachfolgerkante eines Knotens soll mit 0, die rechte Nachfolgerkante mit 1 bezeichnet werden. Ein innerer Knoten hat als Wahrscheinlichkeit die Summe der Wahrscheinlichkeiten seiner beiden Nachfolger. Ein Blatt hat als Wahrscheinlichkeit die Wahrscheinlichkeit des Auftretens des gespeicherten Zeichens.

Implementiert einen Huffman Baum in der Klasse *HuffmanBaum*. Für die Knoten des Baumes soll eine Klasse *HKnoten* verwendet werden. Folgende Methoden der Klasse *HuffmanBaum* sollen (mindestens) implementiert werden:

Aufgabe 1.1 Häufigkeiten

Für die Huffman-Codierung braucht ihr zuerst eine Häufigkeitsverteilung der Zeichen als Schlüssel. Schreibt eine Methode, die die Häufigkeit aller in einem *String* enthaltenen Zeichen bestimmt und diese in einem Array ablegt. Um die Häufigkeit zu bestimmen, könnt ihr einfach zählen, wie oft ein Zeichen in dem Text vorkommt. Dazu soll eine Datei mit Hilfe der Klasse *FileInputStream* eingelesen werden. Auch wenn dieser String in der Theorie aus bis zu 65536 verschiedene Zeichen bestehen kann, beschränken wir uns auf den erweiterten ASCII - Zeichensatz mit dem Code [0..255]. Nutzt aus, dass in Java ein *char* auch immer eine Zahl ist.

Aufgabe 1.2 Huffman-Baum pflanzen

Baut den Huffman-Baum auf:

1. Erzeugt eine Liste von Bäumen mit jeweils nur einem Knoten, in denen die Zeichen und ihre jeweilige Häufigkeit gespeichert sind. Die Liste soll aufsteigend nach der Häufigkeit sortiert sein. Tragt nur Zeichen ein, deren Zeichen-Code mindestens 32 (das Leerzeichen) ist. Das erleichtert die spätere Ausgabe.

2. Entfernt die ersten beiden Bäume aus der Liste und hängt sie unter einen gemeinsamen Elternknoten, so dass der erste Baum der linke Kindknoten wird und der zweite Baum der rechte. Die Wurzel des neuen Baums speichert die Summe der Häufigkeiten seiner Kinder. Fügt den so entstandenen Baum wieder in die Liste ein, so dass die aufsteigende Sortierung erhalten bleibt.

3. Wiederholt Schritt 2, bis nur noch ein Baum in der Liste enthalten ist. Dieser ist der Huffman-Baum.

Aufgabe 1.3 Codetabelle

Erzeugt aus einem Huffman-Baum eine Codetabelle, die die jeweilige Kodierung für jedes Zeichen enthält. Dies geht folgendermaßen:

1. Von der Wurzel ausgehend,

2. wenn ein linker Teilbaum beschritten wird, schreibt eine 1 hinter den bisherigen Code,

3. wenn ein rechter Teilbaum beschritten wird, schreibt eine 0 hinter den bisherigen Code,

4. wenn ihr ein Blatt erreicht, habt ihr den Code gefunden und könnt dem Zeichen, das im Blatt gespeichert ist, den Code zuweisen. Dieser wird in die Codetabelle an die Position des jeweiligen Wert des Zeichens geschrieben. Die Codetabelle mit Indizes i=[0...255] soll eine schnelle Kodierung der ASCII Zeichen in den Huffman Code ermöglichen.

Geebt für Eure Kodierung den Code für die jeweils zehn häufigsten Zeichen aus.

Aufgabe 1.4 Hin- und hercodieren

• Implementiert eine Methode *kodiere()*, die aus einem String die binäre Kodierung als String von Einsen und Nullen auf Basis der erstellten Codetabelle erzeugt (z.B. 'Hallo' → '010110....101').

• Implementiert eine Methode *dekodiere()*. Diese Methode soll aus einem kodierten String einen dekodierten String erstellen.

Aufgabe 1.5 Fragen zum Verfahren

1. Warum ist die Huffman Kodierung präfixfrei?

2. Wie groß ist die Tiefe jedes Blattes im Huffman Baum, wenn alle $N = 2^k$ Zeichen gleichverteilt sind?

3. Wieso erzeugt der oben beschriebene Algorithmus eigentlich für häufige Zeichen kurze Codes und für seltene lange?

4. Warum terminiert der Aufbau des Huffman-Baums?