Praktische Informatik II SS 2009

Übungsblatt 6

Abgabe: 18.05.2009

Aufgabe 1 Das Wissen der Welt...(70%)

. . . steckt in der Wikipedia. Das stimmt zwar noch nicht ganz, trotzdem enthält die Internet-Enzyklopädie eine Unmenge an Wissen, das sogar in zahlreichen Sprachen vorliegt. Dieses Wissen braucht viel Platz. Da wäre es doch schön, wenn man die dort gespeicherten Texte kompakter ablegen könnte. Eine Möglichkeit wäre, die Seiten im sog. Huffman-Code zu speichern, d.h. häufige Zeichen mit kurzen Bitfolgen zu kodieren und seltene mit längeren. Welche Zeichen selten und welche häufig sind, hängt in erster Linie von der verwendeten Sprache ab. Daher wäre es gut, einen Huffman-Code pro Sprache zu ermitteln, der dann zum Kodieren und Dekodieren der Seiten benutzt werden kann. Genau das sollt ihr tun.

Aufgabe 1.1 Häufigkeiten

Für die Huffman-Codierung braucht ihr zuerst die Häufigkeit der Zeichen auf den Webseiten. 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. Da ein String aus Unicode-Zeichen besteht, kann es bis zu 65536 verschiedene Zeichen geben. Nutzt aus, dass in Java ein char auch immer eine Zahl ist und deshalb als Index in einem Array benutzt werden kann.

Aufgabe 1.2 Huffman-Baum

Baut den Huffman-Baum wie in der Vorlesung besprochen auf:

1. Erzeugt aus dem zuvor erstellten Array eine Liste von binären Bäumen mit jeweils nur einem Knoten. In jedem Knoten wird ein Zeichen und dessen Häufigkeit gespeichert. Tragt nur Zeichen ein, deren Häufigkeit größer als Null und deren Zeichen-Code mindestens 32 (das Leerzeichen) ist. Das erleichtert die spätere Ausgabe. Die Liste soll aufsteigend nach der Häufigkeit sortiert werden.
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.

Hinweise. Ihr dürft für die Liste die Klasse *java.util.LinkedList<T>* verwenden. Sie stellt euch u.a. Methoden zum Einfügen und Löschen von Elementen zur Verfügung. Für das sortierte Einfügen müssen eure Knoten bzw. Bäume vergleichbar sein. Nutzt hierfür das Interface *Comparable*.

Aufgabe 1.3 Kodierung

Nun muss noch der Code für die einzelnen Zeichen erstellt werden. Dies geht folgendermaßen:

- Von der Wurzel ausgehend,
- wenn ein linker Teilbaum beschritten wird, schreibt eine 1 hinter den bisherigen Code,
- wenn ein rechter Teilbaum beschritten wird, schreibt eine 0 hinter den bisherigen Code,
- wenn ihr ein Blatt erreicht, habt ihr den Code gefunden und könnt das Zeichen, das im Blatt gespeichert ist, und den Code zurückgeben.

Hinweise. Nutzt die Klasse *java.util.HashMap*, um die Codes für die einzelnen Zeichen abzuspeichern. Dabei soll das jeweilige Zeichen als Hash-Schlüssel dienen, so dass man später zu jedem Zeichen direkt dessen Kodierung in der HashMap nachschlagen kann.

Aufgabe 2 Wissen testen (30%)

Aufgabe 2.1

Stellt eine Abschätzung auf, wie groß die zu erwartende Kompression ausfällt. Könnt ihr euch auch Fälle vorstellen, in denen keine Kompression erreicht werden kann?

Aufgabe 2.2

Lasst zu den nachfolgenden Wikipedia-Seite die Huffman-Kodierung berechnen und gebt für jede dieser Kodierungen den Code für die Zeichen 'a' bis 'z' aus. Begründet kurz das Ergebnis.

- `http://de.wikipedia.org/wiki/Shannon-Fano-Kodierung`
- `http://en.wikipedia.org/wiki/Shannon%E2%80%93Fano_coding`
- `http://es.wikipedia.org/wiki/Algoritmo_de_Huffman`

Hinweise. Zum Einlesen von Wikipedia-Seiten steht auf der PI-2-Webseite ein Archiv mit der Datei *Wikipedia.java* bereit, die eine Methode anbietet, die allen Text einer Seite ausliest und als *String* zurückliefert.