

# Programmierübungen

Wintersemester 2007/2008

## 10. Übungsblatt

18. Januar 2008

Abgabe bis Freitag, 25. Januar 23:59 Uhr

Die Abgabe Ihrer Bearbeitung können Sie im eClaus-System durchführen. Erarbeiten Sie Lösungsideen zu den Aufgaben möglichst in Kleingruppen. Es wird jedoch von Ihnen erwartet, dass jeder Teilnehmer eine eigene Lösung abgibt. Sollten kopierte Quelltexte abgegeben werden, so werden grundsätzlich alle Kopien mit 0 Punkten bewertet. In den Vortragsfolien der Programmierübungen oder im Skript zur Einführung in die Informatik abgedruckte Quelltexte können verwendet werden, müssen aber der Programmierrichtlinie entsprechend formatiert und kommentiert werden.

Beachten Sie die Programmierrichtlinie und kommentieren Sie Ihren Quelltext! Dokumentieren Sie unbedingt Ihre Lösungsidee in den Quelltext-Kommentaren.

<http://www.iste.uni-stuttgart.de/ps/Lehre/WS0708/inf-prokurs>

*Bitte beachten Sie, dass zum Scheinerwerb unter anderem auf den letzten 4 Aufgabenblättern 50% der Punkte erreicht werden müssen. Das letzte Aufgabenblatt wird Nummer 12 sein.*

### Aufgabe 10.1: Huffman-Bäume

(10 Punkte)

Textdateien lassen sich häufig sehr gut komprimieren, da sie nur wenige der verfügbaren Zeichen enthalten.

Ein Zeichen (in Ada-Programmen oft durch den Datentyp `Character` repräsentiert) benötigt meist 1 Byte = 8 Bit Speicherplatz. 8 Bit entsprechen 8 Ziffern des Binärsystems (0 oder 1). Somit lassen sich in einem Byte die Character-Werte mit den Positionsnummern 0 – 255 kodieren.

Um Speicherplatz zu sparen versucht man, häufig verwendeten Zeichen nicht in 8 Bit, sondern durch eine kürzere Kodierung zu speichern. Werden häufig auftretende Zeichen durch einen besonders kurzen Code gespeichert, so ergibt sich eine große Ersparnis. Eine Technik, Kodierungen für Zeichen aufzubauen, sind Huffman-Bäume. Sie haben diese Struktur:

- Huffman-Bäume sind Binärbaume,
- die eine Kante, die von einem Knoten ausgehen kann ist mit 0 beschriftet, die andere mit 1,
- innere Knoten sind unbeschriftet,
- Blätter sind mit einem Zeichen beschriftet,
- zum Aufbau des Baums wird jedem Knoten ein Gewicht (eine ganze Zahl zugeordnet).

Entwerfen Sie geeignete Datentypen zur Darstellung von Huffman-Bäumen.

Schreiben Sie ein Programm `code`, das eine Textdatei einliest und dafür einen Huffman-Baum aufbaut. Der Name der Textdatei soll als erstes Kommandozeilen-Argument übergeben werden können. Verwenden Sie den folgenden Algorithmus:

1. Lies die Datei komplett ein und zähle wie oft jedes Zeichen vorkommt (z. B. 13\* 'a', 14\* 'b', ...).
2. Erzeuge Blattknoten für jedes Zeichen, das mindestens einmal vorkommt. Beschrifte das Blatt mit dem Zeichen und setze sein Gewicht auf die Häufigkeit des Zeichens.
3. Erzeuge eine Liste  $L$  aller Knoten. Enthält  $L$  weniger als 2 Buchstaben, so füge solange neue Blätter hinzu, hinzu bis  $L$  genau 2 Knoten enthält. Beschrifte diese Knoten mit beliebigen Zeichen.
4. Wähle die zwei Knoten von  $L$  mit dem kleinsten Gewicht aus (haben mehrere Knoten das kleinste Gewicht, wähle 2 von diesen willkürlich aus), entferne diese Knoten aus  $L$ .
5. Erzeuge einen neuen inneren Knoten und setze sein Gewicht auf die Summe der Gewichte der beiden gewählten Knoten.
6. Setze die 2 gewählten Knoten als erstes und zweites Kind des neuen Knoten. Hänge den Knoten mit dem kleineren Gewicht an die Kante mit der Beschriftung 0, den mit dem größeren an die Kante mit Beschriftung 1, bei gleichem Gewicht wähle willkürlich.
7. Füge den neuen Knoten zu  $L$  hinzu.
8. Ist in  $L$  genau ein Knoten, so bestimme diesen als Wurzel des fertigen Huffman-Baums und beende diesen Algorithmus.

Verfolgt man den Weg von der Wurzel des Huffman-Baums zu einem Blatt, so ergeben die überquerten Kantenbeschriftungen die Kodierung des Zeichens, das auf dem Blatt steht. Lassen Sie das Programm eine Liste der Zeichen und ihrer Kodierung ausgeben. Diese Liste darf beliebig sortiert sein.

Beispiel:

Eingabedatei `text.txt`:

Programmieruebungen

Ausgabe (der zugehörige Baum wird in Abbildung 1 gezeigt):

```
P: 10110
a: 10111
b: 0000
e: 110
g: 001
i: 0001
m: 010
n: 011
o: 1010
r: 111
u: 100
```



### Aufgabe 10.3: Dekompression

(5 Punkte)

Die Dekompression ist in der Praxis etwas schwieriger, da hierzu natürlich der Huffman-Baum benötigt wird. Diesen Baum kann man anhand der Ausgabe des `compress`-Werkzeugs nicht rekonstruieren. Deshalb ist diese Aufgabe etwas unrealistisch vereinfacht:

Schreiben Sie ein weiteres Programm `decompress`, das aus der als Kommandozeilen-Argument übergebenen Textdatei zunächst einen Huffman-Baum aufbaut. Danach liest das Programm von der Standard-Eingabe einen Text, der nur aus den Zeichen 1, 0 und Leerzeichen besteht. Leerzeichen werden vom Programm ignoriert. Das Programm dekomprimiert den eingegebenen Text und gibt das Resultat am Ende aus.

Beispiel:

Eingabedatei `text.txt`:

Programmieruebungen

Eingabe:

```
10110 111 1010 001 111 10111 010 010 0001 110 111 100 110 0000
100 011 001 110 011
```

Ausgabe:

Programmieruebungen

Hinweise:

- Verwenden Sie Funktionalität aus der ersten Aufgabe (in einem Paket!)
- Sollte Ihre Abgabe zu Aufgabe 1 nicht vollständig sein, so schreiben Sie stattdessen eine Prozedur, die den Baum nicht aus einer Textdatei aufbaut, sondern irgendeinen sinnvollen Baum aufbaut. Die Aufgaben werden separat bewertet.