HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Heinrich-Heine-Universität Düsseldorf
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik, Abteilung Betriebssysteme
Prof. Dr. Michael Schöttner
Kevin Beineke
Ludmila Himmelspach
Stefan Nothaas

11. Übungsblatt „Programmierung“ im Wintersemester 2016/17

Abgabe bis 23. Januar 2017, 07:59 Uhr über das Abgabesystem

Aufgabe 1: Maximumssuche in Binärbäumen — 10 Punkte

Erweitern Sie das Beispiel zum Binärbaum aus der Vorlesung um eine Maximumssuche. Dabei soll die Suche einmal **rekursiv** und einmal **iterativ** implementiert werden. Die `main`-Methode ist entsprechend anzupassen und soll sinnvolle Tests zu Ihren beiden neuen Methoden enthalten.

Aufgabe 2: Höhe von Binärbäumen — 20 Punkte

Erweitern Sie das Beispiel zum Binärbaum aus der Vorlesung um eine Methode, welche die Höhe des Baumes (längster Pfad von einem Blatt zur Wurzel) **rekursiv** bestimmt. Die Höhe eines Baumes, der nur aus einer Wurzel besteht, ist 0. Wenn die Wurzel genau ein Kind hat, beträgt die Länge 1 usw. Die `main`-Methode ist entsprechend anzupassen und soll sinnvolle Tests zu Ihrer Methode enthalten.

Aufgabe 3: Hashtable mit linearem Sondieren — 30 Punkte

Eine Hashfunktion bezeichnet eine Funktion, die von einem großen Definitionsbereich in einen, im allgemeinen deutlich kleineren, Wertebereich abbildet. Die Eingabewerte werden dabei als Schlüssel und die Ausgabewerte als Hashwerte bezeichnet. Diese Funktion lässt sich ebenfalls dazu benutzen, den Speicherindex innerhalb einer Tabelle zu berechnen. Diese Kombination von Tabellenindizierung und Hashwert ist auch als Hashtabelle bekannt.

a) Erklären Sie, was man unter einer Kollision bezüglich einer Hashfunktion versteht.

b) Erklären Sie, was man unter linearem Sondieren versteht.

c) Damit der Hashwert in seiner Größe begrenzt ist, enthält die zugehörige Funktion eine Modulo-Operation. Begründen Sie, warum sich Primzahlen oftmals besser als Modulo-Operator eignen.

d) Implementieren Sie eine Hashtabelle der Größe 67 für Hashing mit linearem Sondieren. Verwenden Sie für die Tabelle die folgende Hashfunktion für einen String der Länge $n > 1$:

$$h(s_n) = (((h(s_{n-1}) << 8) + s_n) \mod k) \qquad \text{mit} \quad k = 67,\ h(s_0) = s_0$$

Dabei sei s_n der ASCII-Code[1] des n-ten Zeichens eines Wortes. Das erste Zeichen eines Wortes ist mit s_0 bezeichnet. Mit $<<$ lassen sich bitweise Verschiebungen nach links ermöglichen. D.h. die Eingabe wird um die angegebene Anzahl an Bits in Richtung des höchstwertigen Bits verschoben und mit Nullen aufgefüllt.

Beispiel für die bitweise Verschiebung: $10001101 << 4 = 11010000$

Beispiel zur Anwendung der Hashfunktion anhand des Strings "abc":

$$h(\text{'}c\text{'}) = (h(\text{'}b\text{'}) << 8) + \text{'}c\text{'}) \mod 67 = (((((h(\text{'}a\text{'}) << 8) + \text{'}b\text{'}) \mod 67) << 8) + \text{'}c\text{'}) \mod 67$$

[1] siehe auch *http://de.wikipedia.org/wiki/ASCII-Tabelle*

$$= (((((\text{'}a\text{'} << 8) + \text{'}b\text{'}) \mod 67) << 8) + \text{'}c\text{'}) \mod 67$$

$$= ((((97 << 8) + 98) \mod 67) << 8) + 99) \mod 67$$

$$= 27$$

e) Fügen Sie nacheinander folgende 35 Wörter in die Tabelle ein, lassen Sie sich nach jedem Einfügevorgang die Kollisionsrate (Gesamtanzahl bisher aufgetretener Kollisionen in Verbindung mit aktuellem Füllstand der Tabelle) ausgeben und geben Sie eine Tabelle mit allen Iterationen an. Orientieren Sie sich bei der Ausgabe an der unten stehenden Tabelle. Hinweis: Der Füllstand der Tabelle bezieht sich nur auf die belegten Einträge der Tabelle.

Bleistift	Pflanze	Fussballer	Regenschirm	Kartographie
spielen	Fahrrad	Strasse	Weltreise	Schloss
fahren	Stadt	Kaufhaus	fliegen	Golf
Gewitter	Erholung	erholen	Klingel	Wetterhahn
Urlaub	Hund	einkaufen	Wasser	schlendern
Feldweg	traeumen	Pizzeria	Kurierfahrer	Zoo
Buch	Yacht	Papagei	reisen	Baum

Wortnr.	Füllgrad	Kollisionen	Kollisionen / Füllrate
1	1 (1%)	0	0.00
2	2 (3%)	1	0.50
3	3 (4%)	1	0.33
...	...	...	...

f) Bewerten Sie die Hashfunktion anhand der Kollisionsrate.