Prof. Dr. R. Schrader
A. Schulze

Probeklausur zur Informatik I

Teil A – Allgemeiner Teil

Aufgabe A-1: ($\mathcal{O}$-Notation) **6 Punkte**

Sei $n \in \mathbb{N}$. Bestimmen Sie für die folgenden Funktionen f_i, $i = 1, 2, 3$, möglichst langsam wachsende Funktionen g_i, $i = 1, 2, 3$, so dass $f_i \in \mathcal{O}(g_i)$.

$f_1(n) = 12$,
$f_2(n) = \lfloor n^{0.5} \rfloor$ ($\lfloor x \rfloor$ ist die größte ganze Zahl kleiner gleich x),
$f_3(n) = 6\,n^2 + 5\,n\,log(n)$.

Aufgabe A-2: (Quicksort) **6 Punkte**

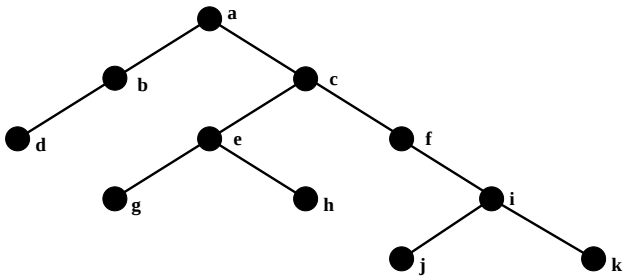
Zeichnen Sie den Rekursionsbaum, der entsteht, wenn die Eingabefolge

$$A = [5, 6, 9, 12, 8, 10, 3, 11, 4, 7]$$

mit dem Algorithmus „Quicksort“ sortiert wird.

Aufgabe A-3: (Binärbäume) **3 + 2 Punkte**

(a) Geben Sie die Präordnung, Postordnung und die symmetrische Durchmusterung des folgenden Baumes an.



(b) Angenommen, Sie kennen die Prä- und Postordnung eines binären Baumes. Können Sie diesen rekonstruieren? Begründen Sie Ihre Aussage.

Aufgabe A-4: (Suchen) **4 Punkte**

Gegeben sei ein Feld $A = [5, 8, 9, 11, 22, 24, 26, 35, 36]$. Beschreiben Sie die Suche nach dem Schlüssel 27 im obigen Feld A durch die Angabe der Folge der ausgeführten Schlüsselvergleiche, wenn als Suchstrategie

(a) lineare Suche

(b) binäre Suche

gewählt wird.

Aufgabe A-5: (Fibonacci-Zahlen) **6 Punkte**

Beweisen Sie, dass für die m-te Fibonacci-Zahl F_m gilt:

$$F_m = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^m - \left(\frac{1-\sqrt{5}}{2} \right)^m \right]$$

Aufgabe A-6: (AVL-Bäume) **6 Punkte**

(a) Definieren Sie den Begriff AVL-Baum.

(b) Beschreiben Sie ein Verfahren, mit dem man testen kann, ob ein vorgegebener binärer Suchbaum ein AVL-Baum ist. Geben Sie die Laufzeit Ihres Verfahrens an.

Aufgabe A-7: (B-Bäume) **2 + 5 Punkte**

(a) Definieren Sie den Begriffe (a,b)-Baum.

(b) Gegeben ist ein Baum, der anfänglich aus einer Wurzel mit Schlüssel 13 und zwei Blättern, eins mit Schlüsseln 10 und das andere mit Schlüssel 14 besteht. Geben Sie die (3,5)-Bäume an, die durch sukzessives Einfügen der Schlüssel 15, 11, 4, 8, 7, 3, 2, 13, 32, 20 in dieser Reihenfolge in den Baum entstehen.

Teil B – Programmierteil

Achten Sie in Ihren Programmen auf gute Lesbarkeit durch ausführliche Kommentare, erkennbare Gliederung und geeignete Benennung der Variablen.

Aufgabe B-1: **15 Punkte**
Hashing

In der Vorlesung haben Sie das Hashing kennen gelernt. Die Klasse `Hash` soll eine Tabelle enthalten, die m Integer-Zahlen Platz bietet. Diese werden mit ebenfalls ganzzahligen und **positiven** Schlüsseln mit Hilfe der Divisionsmethode eingefügt. Behandeln Sie Kollisionen durch Verkettung, jedoch nicht mittels einer verketteten Liste: Zur Vereinfachung der Implementierung nehmen Sie an, dass höchstens m Elemente eingefügt werden sollen und dass die verwendete Datenstruktur nicht speichereffizient sein muss (z.B.: zweidimensionales Integer-Array).
Implementieren Sie (schreiben Sie als Kompilierbaren Java- oder C++-Code) die folgende Klasse `Hash` mit den angegebenen Methoden für Integer-Werte mit Hilfe eines/mehrerer Integer-Arrays:

```
public class Hash
{
//Konstruktor
//Parameter m:
//Gibt die Groesse der Hash-Tabelle an.
public Hash(int m){}

//Einfügen des Elementes zahl
//mit dem Schluessel s
public void hash(int zahl, int s){}

//Ausgeben eines Elementes
//mit Schluessel s
public vector<int> lookup(int s){}

//Löschen eines Elementes
//mit Schluessel s
public void delete(int zahl, int s){}
}
```

Hinweis: Es sollen in die Hash-Tabelle Zahlen (`zahl`) eingefügt werden, jedoch mit einem Schlüsselwert (`s`), der sich von der Zahl selbst unterscheidet. (Schlüssel `s` ist also nicht der berechnete Hashschlüssel, sondern der Ausgangswert für die Hashfunktion.)

Aufgabe B-2: **15 Punkte**
Radix-Sortierung

In der Vorlesung wurde neben den allgemeinen Sortierverfahren das spezielle Verfahren Radix-Sortierung vorgestellt: Es sollen n ganzzahlige, i-stellige Schlüssel im Dezimalsystem sortiert werden. Dazu werden alle Schluessel nach der aktuellen Ziffer in die Fächer $F_0, \ldots, F_9$ verteilt, beginnend mit der i-ten Ziffer. Nach der Verteilung werden die Schluessel in Reihenfolge der Fächer „aufeinander gelegt“ und erneut in Fächer sortiert, nun nach der $(i-1)$ ten Stelle. Nach Abschluss von i Runden sind die Schlüssel sortiert.
Implementieren Sie (schreiben Sie als compilierbaren Java- oder C++-Code) die Funktion `sort` der folgenden Klasse `RadixSort`. Diese soll das übergebene Integer-Array `field`, das n ganzzahlige, i-stellige Schlüssel im Dezimalsystem enthält, durch i-maliges einsortieren in Fächer sortieren.
Hinweis: Verwenden Sie für das Sortieren in Fächer die Klasse `Hash`.

```
public class RadixSort
{
public void sort(int n, int i, int field[]){}
}
```