Jessica

HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

Prof. Dr. W. Rülling
WS 2015/16

1/3 bestehen
2/3 1
Taschenrechner!

Klausur
Algorithmen und Datenstrukturen
(AIB2, CNB2)

Name	Vorname	Matrikelnummer	Semester

Nr	Thema	erreichbare Punkte	erreichte Punkte
1	Obere und untere Schranken	9	
2	Balancierte Bäume	9	
3	Floyd-Warshall Algorithmus	8	
4	Karatsuba-Multiplikation	10	
5	Flussproblem	10	
6	Allgemeines	7	
7	Wortvergleich	8	
8	Klasse LinkedList	12	
9	Sortierung von Geraden	9	
	Summe	82	

1. Überprüfen Sie die Aufgabenblätter anhand obiger Liste auf Vollständigkeit.
2. Tragen Sie bitte oben auf diesem Deckblatt Ihren Namen, Ihr Semester und die Matrikelnummer ein. Nach der Klausur geben Sie bitte das Deckblatt zusammen mit Ihren Lösungsblättern ab. Die Aufgabenblätter können Sie behalten.
3. Versehen Sie jedes abgegebene Lösungsblatt oben links mit Ihrem Namen.

Aufgabe 1: Obere und untere Schranken

a) Geben Sie für die folgenden Funktionen jeweils möglichst gute größenordnungsmäßige obere Schranken (O-Notation) und untere Schranken (Ω-Notation) an.

$$
\begin{aligned}
f(n) &= \left(n^2 + 3n\right) \cdot \left(2n^2 + 4\right) \\
g(n) &= \begin{cases} 8 + \left(\frac{n-4}{2}\right) & \text{für ungerade } n > 4 \\ 8 + \log_2(n^2 - 16) & \text{für gerade } n > 4 \\ \frac{1}{2} \cdot n^2 & \text{für } n \leq 4 \end{cases} \\
h(n) &= 3\log_3 n + 2\sqrt{4n}
\end{aligned}
$$

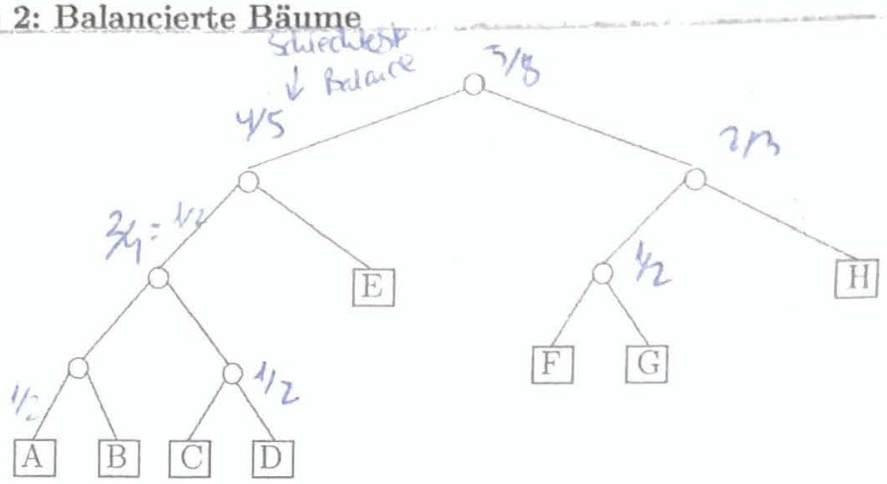
b) Bestimmen Sie eine möglichst gute obere Schranke für die größenordnungsmäßige Laufzeit $T_y(n)$ einer Funktion y, wenn bereits folgender rekursiver Ansatz für T_y gegeben ist

$$
T_y(n) = \begin{cases} 3 & \text{für } n <= 1 \\ 2 \cdot T_y(\frac{n}{3}) + 4n & \text{für } n > 1 \end{cases}
$$

c) Schätzen Sie die Laufzeit T_z der angegebenen (rekursiven) Java-Methode `int z(int n)` größenordnungsmäßig nach oben ab.

```
int z(int n)
{if (n>10)
   return z(n/3) + z((n-2)/3);
 else
   return n;
}
```

Aufgabe 2: Balancierte Bäume



a) Geben Sie für den angegebenen Baum eines Wörterbuchs für alle inneren Knoten die Balancewerte an.

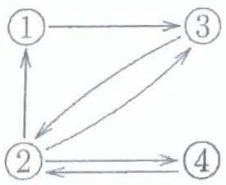
b) Geben Sie den α-Wert für den Baum an.

c) Rebalancieren Sie den Baum am Knoten mit der schlechtesten Balance, um den α-Wert zu verbessern und geben Sie an, welche Rotationsart Sie dabei verwenden.

d) Welche größenordnungsmäßige Laufzeit braucht man, um n Daten in einen anfangs leeren Baum einzutragen?

e) Welche größenordnungsmäßige Laufzeit braucht eine Doppelrotation?

Aufgabe 3: Floyd-Warshall Algorithmus



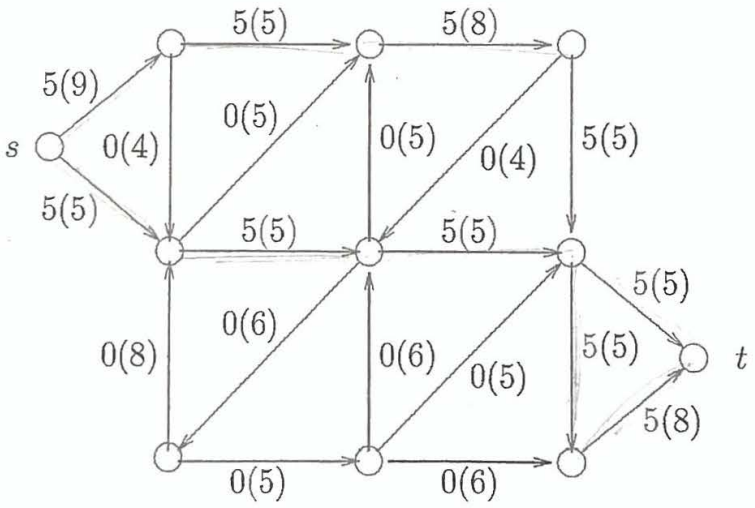
Ermitteln Sie mit Hilfe des Floyd-Warshall Algorithmus die Distanzen zwischen je zwei Knoten des angegebenen Graphen. Geben Sie dabei alle während des Algorithmus entstehenden Matrizen an.

Aufgabe 4: Karatsuba-Multiplikation

a) Berechnen Sie nach der Karatsuba-Methode den Wert a^2 für die 6-stellige Binärzahl $a = 100101$ indem Sie die 6-stellige Multiplikation auf drei kleinere Multiplikationen zurückführen. Geben Sie alle dabei entstehenden Zwischenergebnisse an.
(Die kleineren Multiplikationen müssen nicht weiter zerlegt werden.)

b) Berechnen Sie nach der Karatsuba-Methode das Produkt $a \cdot b$ für die 4-stelligen Dezimalzahlen $a = 2015$ und $b = 2016$ und geben Sie alle dabei entstehenden Zwischenergebnisse an.
(2- und 3-stellige Zahlen müssen nicht weiter zerlegt werden).

Aufgabe 5: Flussproblem



Die obige Skizze zeigt einen gerichteten Graphen mit Kantenkapazitäten (Kapazitäten in Klammern). Zusätzlich ist ein Fluss von s nach t der Stärke 10 eingetragen.

a) Erstellen Sie den Hilfsgraphen der möglichen Fluss-Verbesserungen

b) Konstruieren Sie mit dem Hilfsgraphen einen verbesserten Fluss und geben Sie seine Stärke an. Skizzieren Sie den verbesserten Fluss.

c) Skizzieren Sie einen minimalen Schnitt für Flüsse von s nach t und geben Sie seine Kapazität an.

Aufgabe 6: Allgemeines

a) Was versteht man unter einer Klassenvariablen?
b) Wann nennt man eine Klasse **abstrakt**?
c) Was versteht man unter einer **inneren** Klasse?
d) Was bewirkt der Methodenaufruf `super()`?
e) Welche Bedeutung hat das Schlüsselwort **throws** im Kopf einer Methode?
f) Welche größenordnungsmäßige Laufzeit (O-Notation) braucht die Methode `compareTo`, wenn zwei Strings der Länge n verglichen werden müssen?
g) Welche größenordnungsmäßige Laufzeit (O-Notation) braucht ein guter Algorithmus für die Multiplikation zweier `BigInteger`-Zahlen der Länge n Bit?

Aufgabe 7: Wortvergleich

a) Schreiben Sie eine Methode `boolean test(String sa,String sb)`, die für zwei Strings `sa` und `sb` prüft, ob in beiden Strings genau die gleichen Zeichen vorkommen.

   Hinweis: Tragen Sie dazu die Zeichen aus `sa` in eine Menge `a` und die Zeichen aus `sb` in eine Menge `b` ein und testen Sie dann mit der Methode `equals` des Interface `Collection` ob beide Mengen gleich sind.

b) Benutzt man in a) statt der Methode `equals` den Operator ==, erhält man nicht das gewünschte Ergebnis. Woran liegt das?

c) Welche größenordnungsmäßige Laufzeit $T(n)$ hat Ihre Implementierung aus a), wenn die beiden Strings die gleiche Länge n haben?

Aufgabe 8: Klasse LinkedList

Im folgenden ist eine noch leere Klasse `Hilfsfunktionen` skizziert, die mit einem Typ `T` parametrisiert ist.

```
public class Hilfsfunktionen<T extends Comparable<T>> {

}
```

Für diese Klasse sollen Sie zwei Methoden programmieren:

a) Schreiben Sie eine Java-Methode `public T minimum(LinkedList<T> x)` die den kleinsten Eintrag in der Liste `x` als Ergebnis zurückliefert.

b) Schreiben Sie eine Methode `public boolean monoton_fallend(LinkedList<T> x)` die prüft, ob die Einträge in in der Liste `x` nach absteigenden Werten sortiert sind.

Im folgenden Hauptprogramm einer Klasse `Beispiel` soll die obige Methode `minimum` ausprobiert werden. Dabei liefert der Java-Compiler allerdings bei der Ausgabeanweisung eine Fehlermeldung.

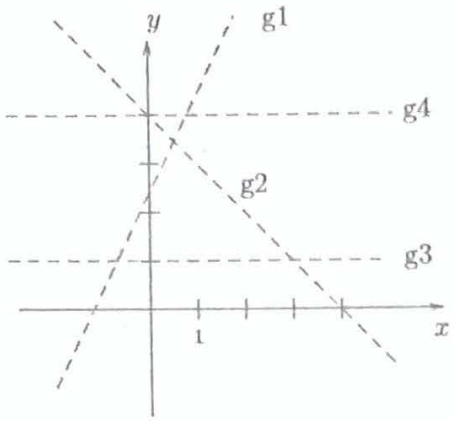
```
public class Beispiel {
   public static void main(String[] args) {
      LinkedList<Float> x= new LinkedList<Float>();
      x.add(3.4); x.add(2.5); x.add(9.15); x.add(0.8);
      System.out.println( "Das Minimum von x beträgt " + minimum(a));  // FEHLER!
    }
};
```

c) Geben Sie eine korrekte Schreibweise für den Aufruf der Methode `minimum` im angegebenen Programm an. Beachten Sie dabei, dass die Methoden nicht als `static` deklariert sind!

d) Wenn man in der Klasse `Hilfsfunktionen` versucht, die Methoden als `static` zu deklarieren, erhält man ebenfalls eine Fehlermeldung. Woran liegt das?

Aufgabe 9: Sortierung von Geraden

Die folgende Klasse Gerade soll Geraden $y = ax + b$ in der reellen Ebene darstellen. Erweitern Sie das gegebene Hauptprogramm um eine sortierte Ausgabe der Geraden. Dabei sollen die Geraden primär nach ihrem b-Wert und bei gleichen b-Werten nach der Steigung a sortiert ausgegeben werden. Im angegebenen Beispiel wäre die Sortierung: g3 g1 g2 g4.



Hinweis: Programmieren Sie die noch fehlende compareTo-Methode und tragen Sie im Hauptprogramm alle Geraden aus v in eine sortierte Menge ein, die Sie anschließend einfach ausgeben können.

```
import java.util.*;

public class Gerade implements Comparable<Gerade>
{ private String name;
  private Double a; // Steigung
  private Double b; // y-Achsenabschnitt

  public Gerade(String s, double a, double b)
   {name=s; this.a=a; this.b=b;
   }

  String toString() {return name;}

  // fehlende compareTo-Methode !

  public static void main(String[] args)
  {Gerade[] v= {new Gerade("g1",  2,  1.5),
                new Gerade("g2", -1,  4),
                new Gerade("g3",  0,  1),
                new Gerade("g4",  0,  4)};

   // Sortierte Ausgabe der Geraden aus v fehlt noch!

  }
}
```