Klausur Algorithmen und Datenstrukturen

Sommersemester 2018

6. April 2018

Ihre Matrikelnummer:

Achtung: Geben Sie keine weiteren persönlichen Daten außer der Matrikelnummer an!

Generelle Hinweise:

- Schreiben Sie **lesbar** (lesbar für die *Korrektoren*, nicht nur lesbar für *Sie*!).
- Schreiben Sie zu Ihrer Sicherheit die Matrikelnummer auf jedem Blatt in das jeweils dafür vorgesehene Feld.
- Geben Sie in Ihrer Lösung zu jeder Aufgabe an, welcher Text zu welcher Teilaufgabe gehört (wenn Sie Ihre Lösungen zu den Teilaufgaben strikt nacheinander, also ohne Hin- und Hersprünge zwischen Teilaufgaben verfassen, reicht es natürlich, die jeweilige Teilaufgabe jeweils zu Beginn anzugeben).
- Sie können selbstverständlich auch Zwischen- und Nebenergebnisse in Ihre Abgabe schreiben, um leichter zum Ergebnis zu kommen. Stellen Sie dann aber die eigentliche Lösung klar heraus.

Punkteverteilung:

Ausgabe	1	2	3	4	5	6	7	Summe
Punktzahl	13	7	10	15	15	20	20	100

Aufgabe 1: 13 Punkte

Aus der Vorlesung kennen Sie den Algorithmus Pivot Partitioning. Sei nun die folgende Sequenz S gegeben:

Index	1	2	3	4	5	6	7	8	9	10
Wert	12	22	26	27	14	27	19	26	28	24

Index	11	12	13	14	15	16	17	18	19	20
Wert	21	18	22	10	14	22	25	21	15	15

Das Pivotelement ist $p := 14$. Geben Sie die Sequenz S nach $i := 2$ an (also nach zwei Iterationen / Induktionsschritten).

Ihr Ergebnis:

Index	1	2	3	4	5	6	7
Wert							

Index	8	9	10	11	12	13	14
Wert							

Index	15	16	17	18	19	20
Wert						

Zeiger $i1$:

Zeiger $i2$:

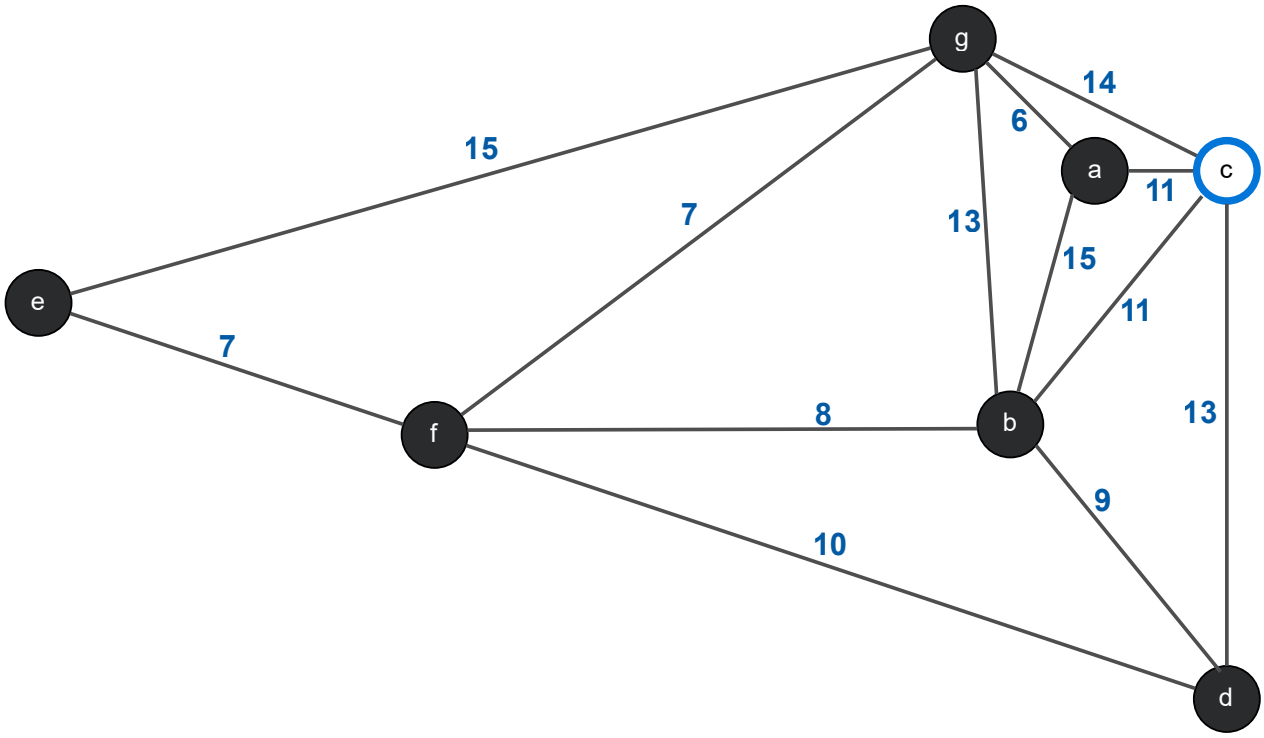
Zeiger $i3$:

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Freie Seite für Ihre Nebenrechnungen (werden nicht gewertet):

Aufgabe 2: 7 Punkte

Aus der Vorlesung kennen Sie den Algorithmus *Dijkstra*.
Gegeben sei der folgende Graph:



Geben Sie ausgehend vom Startknoten n = c die Entfernungen zum Zeitpunkt i = 6 an (also nach sechs Iterationen / Induktionsschritten).

Ihr Ergebnis:

a	b	c	d	e	f	g

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Freie Seite für Ihre Nebenrechnungen (werden nicht gewertet):

Aufgabe 3: 10 Punkte

Aus der Vorlesung kennen Sie *double hashing*.

Hashfunktion:

$$F(i, 18, K) \;=\; (F_1(18, K) + (i-1) \cdot F_2(18, K))$$
$$F_1(N_{\max}, K) \;=\; (K \bmod N_{\max})$$
$$F_2(N_{\max}, K) \;=\; K \;+\; (K \bmod N_{\max})$$

Fügen Sie die folgenden Keys in dieser Reihenfolge in die Hashtabelle ein:

$K_1 = 14$, $K_2 = 4$, $K_3 = 10$, $K_4 = 27$, $K_5 = 23$, $K_6 = 2$, $K_7 = 8$,
$K_8 = 7$, $K_9 = 3$, $K_{10} = 1$, $K_{11} = 6$, $K_{12} = 26$

Ihr Ergebnis:

0	1	2	3	4	5

6	7	8	9	10	11

12	13	14	15	16	17

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Freie Seite für Ihre Nebenrechnungen (werden nicht gewertet):

Aufgabe 4: 15 Punkte

Aus der Vorlesung und vom Java-Übungsblatt kennen Sie lineare Listen und diese Klasse für Listenelemente:

```
public class ListItem <T>  {
     public T            key;
     public ListItem<T>  next;
}
```

Konkrete Aufgabe: Schreiben Sie eine Methode

```
int numberOfSections ( ListItem<T> list, T sep, Comparator<T> cmp )
```

Die Methode darf ohne Überprüfung davon ausgehen, dass `list` auf den Kopf einer korrekt gebildeten Liste verweist, die auch leer sein kann (also `list == null` ist möglich) dass der Separator `sep != null` ist und `cmp` auf ein Objekt einer Klasse verweist, die `Comparator<T>` implementiert. Die Methode soll zählen, wie viele Abschnitte die Liste enthält, in denen `sep` **nicht** vorkommt.

Beispiel: Liste von Character, Leerzeichen als Separator:

$$(\text{␣ab␣c␣␣def␣g␣␣␣hijk␣l␣m␣␣n␣op␣␣}) \longrightarrow 9$$

Verbindliche Anforderungen:

- Die Methode muss *iterativ* sein, das heißt, Rekursion ist nicht erlaubt.
- Neben der Methode `compare` von `Comparator<T>` darf keine Funktionalität aus der Standardbibliothek von Java oder anderen Bibliotheken verwendet werden.
- Die asymptotische Komplexität muss insgesamt linear bleiben.

Erinnerung: Methode `compare` von `Comparator<T>` liefert $+1$ (bzw. -1) zurück, falls der erste Parameter größer (bzw. kleiner) als der zweite ist, bei Gleichheit 0.

Ihre Lösung schreiben Sie auf die folgenden Seiten bis zur nächsten Aufgabe:

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Aufgabe 5: 15 Punkte

Aus der Vorlesung und vom Java-Übungsblatt kennen Sie ternäre Bäume. Betrachten Sie folgende Klasse für Baumknoten:

```
public class TernaryTreeNode<T> {
    public T key1;
    public T key2;
    public TernaryTreeNode<T> left;
    public TernaryTreeNode<T> middle;
    public TernaryTreeNode<T> right;
}
```

Konkrete Aufgabe: Schreiben Sie eine Methode

```
void makeBinary ( TernaryTreeNode<T> root )
```

So wie Sie es kennen, wird durch die Attribute `left`, `middle` und `right` ein ternärer Baum gebildet (der auch leer sein kann, das heißt, `root==null` ist möglich). Die Methode darf davon ausgehen, dass dieser Baum korrekt gebildet ist und die Suchbaumeigenschaft nach einer unbekannten (und für die Aufgabe auch irrelevanten) Ordnung auf `T` erfüllt. Insbesondere liegt `key1` zwischen den Schlüsselwerten im linken und mittleren und `key2` – falls ungleich `null` – zwischen denen im mittleren und rechten Teilbaum.

Die Methode `makeBinary` soll den Eingabebaum so umstruktuieren, dass `node.right == null` und `node.key2 == null` für alle Knoten im Baum ist. Dabei sollen keine Schlüsselwerte verlorengehen oder hinzukommen, und die Suchbaumeigenschaft soll nach Anwendung von `makeBinary` mit derselben Ordnung auf `T` wieder erfüllt sein.

Verbindliche Anforderungen:

- Die Aufgabe soll vollständig durch Rekursion gelöst werden, das heißt, Schleifen sind nicht zulässig.
- Es darf keine Funktionalität aus der Standardbibliothek von Java oder anderen Bibliotheken verwendet werden.
- Es darf keine weitere Methode neben der oben spezifizierten Methode `makeBinary` implementiert werden.
- Die asymptotische Komplexität muss insgesamt linear in der Anzahl der Baumknoten bleiben.

Ihre Lösung schreiben Sie auf die folgenden Seiten bis zur nächsten Aufgabe:

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Aufgabe 6: 20 Punkte

Intuitiv gesprochen, berechnet `sheldon` den maximalen
Abstand eines Eintrags ungleich 0 von der Hauptdiagonalen:

```
01        public int sheldon ( long[][] a )  {
02           return leonard ( a, 0 );
03        }

04        public int leonard ( long[][] a, int i ) {
05             if ( i == a.length )
06                return -1;
07             int max1 = howard ( a[i], i );
08             int max2 = leonard ( a, i+1 );
09             if ( max1 > max2 )
10                return max1;
11             return max2;
12        }

13        public int howard ( long[] b, int i ) {
14             int k = -1;
15             for ( int j = 0; j < b.length; j++ ) {
16                 if ( b[j] != 0 && k < i - j )
17                     k = i - j;
18                 if ( b[j] != 0 && k < j - i )
19                     k = j - i;
20             }
21             return k;
22         }
```

Beachten Sie dabei, dass die einzelnen Arrays `a[`i`]` nicht unbedingt gleiche Länge haben müssen. Sie können aber davon ausgehen, dass `a` selbst und auch jedes einzelne Array `a[`i`]` ungleich `null` ist.

Konkrete Aufgaben (a) – (d):

(a) Formulieren Sie **kurz und bündig, aber präzise und unmissverständlich** den Output jeder der Methoden `sheldon`, `leonard` und `howard` (bei `sheldon` also eine Präzisierung der intuitiven Umschreibung vor dem Code). **6 Punkte**

Hinweise: Nennen Sie den Rückgabewert jeweils r und formulieren Sie so etwas wie „die maximale ganze Zahl r, so dass ...“. Ihre Formulierung ist erst dann präzise, wenn sie die Terme `a.length` und `a[`i`].length` bzw. `b.length` geeignet enthält. Für `leonard` ist ein weiterer Index h zweckmäßig, der die Indizes von `a` durchläuft, da i hier schon eine andere Rolle hat.

(b) Formulieren Sie **kurz und bündig, aber präzise und unmissverständlich** die Invariante der Schleife in `howard` sowie den Korrektheitsbeweis (also Induktionsanfang und Induktionsschritt) für diese Invariante. Arbeiten Sie die Induktionsvoraussetzung *explizit* in den Induktionsschritt ein. **4 Punkte**

Hinweise: Beginnen Sie die Invariante analog zum Wiki mit „Nach $h \geq 0$ Iterationen gilt:“. Benennen Sie die Indizes i, j und h überall *explizit*, wo diese von Bedeutung sind.

(c) Formulieren Sie **kurz und bündig, aber präzise und unmissverständlich** die Invariante der Rekursion in `leonard` sowie den Korrektheitsbeweis (also Induktionsanfang und Induktionsschritt) für diese Invariante. Arbeiten Sie wieder die Induktionsvoraussetzung *explizit* ein. **5 Punkte**

Hinweise: Für Invariante und Induktionsanfang machen Sie sich klar, dass der Wert `-1` bei nichtnegativen Zahlen als neutrales Element der Maximumsbildung verwendet werden kann. Beim Induktionsschritt leiten Sie die Werte von `max1` und `max2` her und kombinieren diese beiden Teilergebnisse zu einer Gesamtaussage.

(d) Sei n die Länge von `a` und m das Maximum aus `a[0].length, a[1].length,...,` `a[`$n-1$`].length`. Geben Sie an, unter welchen Umständen der Best Case bzw. der Worst Case bei festgehaltenen Werten n und m eintritt, und begründen Sie, warum die asymptotische Gesamtkomplexität der Methode `sheldon` (inklusive aller Aufrufe von `leonard` und `howard`) im Best Case in $\Theta(n+m)$ und im Worst Case in $\Theta(n \cdot m)$ ist. Geben Sie die Nummern der Zeilen an, die durch ihre wiederholte Ausführung den höchsten Beitrag zur asymptotischen Gesamtkomplexität leisten. **5 Punkte**

Ihre Lösung schreiben Sie auf die folgenden Seiten bis zur nächsten Aufgabe:

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Aufgabe 7: 20 Punkte

(a) Gegeben seien die Funktionen f und g durch $f(x) = 2^x \cdot x^2$ und $g(x) = 3^x$ für alle $x \in \mathbb{R}_0^+$. Beweisen Sie mathematisch, dass g asymptotisch schneller wächst als f (*Hinweise*: Ableitung von a^x ist $\ln_a \cdot a^x$; es gilt $\ln(a/b) = \ln(a) - \ln(b)$; fassen Sie als erstes 2^x und 3^x zusammen). **5 Punkte**

***Verbindliche Anforderung*:** durch mindestens zweimalige Anwendung der Regel von l'Hopital.

(b) Gegeben seien die Funktionen f und g durch $f(x, y) = x^2 \cdot \ln(y)$ und $g(x, y) = y^2 \cdot \ln(x)$ für alle x, $y \in \mathbb{R}_0^+$, (Ableitungsfunktion von ln: $x \to 1/x$). Beweisen Sie mathematisch, dass f und g asymptotisch nicht vergleichbar sind. **4 Punkte**

Unverbindiche Hinweise: Betrachten Sie die Fälle $x = y^2$ und $y = x^2$. Erinnern Sie sich an die Regel $\ln(a \cdot b) = \ln(a) + \ln(b)$.

(c) *Zu Hashtabellen*: In einer anfangs leeren Hashtabelle der Länge N wollen Sie mit Linear Probing insgesamt $n < N$ Elemente einspeichern. Nennen Sie den Worst-Case-Fall, geben Sie die asymptotische Komplexität für diesen Fall an und begründen Sie Ihre Angabe. Gehen Sie davon aus, dass die Hashfunktion zur Berechnung der ersten Adresse konstante Komplexität hat. **3 Punkte**

(d) *Zu Dijkstra*: Warum kann man im Falle, dass einige Kanten negative Längen haben, nicht einfach kürzeste Pfade mit dem Algorithmus von Dijkstra berechnen, indem man zuerst auf jede Kantenlänge das Minimum aller Kantenlängen draufaddiert, so dass alle Kantenlängen nunmehr nichtnegativ sind? Konstruieren Sie dafür ein Gegenbeispiel und begründen Sie, warum dies ein Gegenbeispiel ist. Betrachten Sie dafür *explizit* die von Ihnen zur Begründung verwendeten Pfade und ihre Längen vor und nach der Addition. **3 Punkte**

(e) Beim *Hamiltonkreisproblem* ist ein ungerichteter Graph $G = (V, E)$ gegeben, und gesucht ist die Antwort auf die Frage, ob G einen geschlossenen Pfad (also Kreis) enthält, der jeden Knoten in V genau einmal besucht. Dieses Problem ist $\mathcal{NP}$-vollständig. Leiten Sie daraus ab, dass das verallgemeinerte TSP $\mathcal{NP}$-vollständig ist. „Verallgemeinert" heißt dabei: Von einem Punkt zum anderen kann die Distanz eine beliebige reelle Zahl sein. Führen Sie das Prinzip der polynomiellen Reduktion explizit durch, auch wenn die einzelnen zu berücksichtigenden Punkte offensichtlich zu sein scheinen. **5 Punkte**

Unverbindlicher Hinweis: Zu einer Instanz des Hamiltonkreisproblems definieren Sie eine Instanz des TSP, bei der alle Kanten Länge 0 oder 1 haben.

Ihre Lösung schreiben Sie auf die folgenden leeren Seiten:

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Diese Seite ist nur von den Korrektoren auszufüllen!

1																	
2																	
3																	
4	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O		
5	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O		
6(a)	A	B	C	D	E	F											
6(b)	A	B	C	D													
6(c)	A	B	C	D	E												
6(d)	A	B	C	D	E												
7(a)	A	B	C	D	E												
7(b)	A	B	C	D													
7(c)	A	B	C														
7(d)	A	B	C														
7(e)	A	B	C	D	E												

Vorletzte Spalte: Summe Punkte Teilaufgabe

Letzte Spalte: Summe Punkte Aufgabe

Unterstes Feld: Gesamtpunktzahl aus allen Aufgaben