Prof. Dr. Ir. Joost-Pieter Katoen

Christian Dehnert, Friedrich Gretz, Benjamin Kaminski, Thomas Ströder

# Erste Klausur Datenstrukturen und Algorithmen SS 2015

**Vorname:** _____

**Nachname:** _____

**Matrikelnummer:** _____

**Studiengang** (bitte **genau** einen markieren):

- ○ Informatik Bachelor
- ○ Informatik Lehramt (Bachelor)
- ○ Sonstiges: _____
- ○ Mathematik Bachelor
- ○ CES Bachelor

| | Anzahl Punkte | Erreichte Punkte |
|---|---|---|
| **Aufgabe 1** | **21** | |
| **Aufgabe 2** | **26** | |
| **Aufgabe 3** | **12** | |
| **Aufgabe 4** | **6** | |
| **Aufgabe 5** | **17** | |
| **Aufgabe 6** | **23** | |
| **Aufgabe 7** | **15** | |
| **Summe** | **120** | |

## Allgemeine Hinweise:

- **Auf alle Blätter** (inklusive zusätzliche Blätter) müssen Sie **Ihren Vornamen, Ihren Nachnamen und Ihre Matrikelnummer** schreiben.
- Geben Sie Ihre Antworten in lesbarer und verständlicher Form an.
- Schreiben Sie mit **dokumentenechten** Stiften, nicht mit roten oder grünen Stiften und nicht mit Bleistiften.
- Bitte beantworten Sie die Aufgaben auf den Aufgabenblättern.
- Geben Sie für jede Aufgabe **maximal eine** Lösung an. Streichen Sie alles andere durch. Andernfalls werden alle Lösungen der Aufgabe mit **0 Punkten** bewertet.
- Werden **Täuschungsversuche** beobachtet, so wird die Klausur mit **0 Punkten** bewertet.
- Geben Sie am Ende der Klausur **alle Blätter zusammen mit den Aufgabenblättern ab**.

## Aufgabe 1 ($\mathcal{O}$-Notation): (15 + 6 = 21 Punkte)

**a)** Tragen Sie in die durch ____ gekennzeichneten freien Felder entweder $o$, $\omega$ oder $\Theta$ ein, sodass die entsprechende Aussage gilt. Beispielsweise wäre bei der Aussage $f(n) \in \_\_\_\_(f(n))$ ein $\Theta$ einzutragen, da die Aussage $f(n) \in \Theta(f(n))$ gilt, die Aussagen $f(n) \in o(f(n))$ und $f(n) \in \omega(f(n))$ jedoch beide nicht gelten.

- $n \in \_\_\_\_ \left(\sum_{i=0}^{n} \frac{i}{n}\right)$
- $n^n \in \_\_\_\_ (n!)$
- $\sum_{i=0}^{n} \frac{i}{n} \in \_\_\_\_ (n^2)$
- $n! \in \_\_\_\_ \left(\sum_{i=0}^{3} i^n\right)$
- $\frac{\log_{14}(n)}{n} \in \_\_\_\_ \left(\frac{1}{4}\right)$
- $n^{155} + 3^n \in \_\_\_\_ \left(\sum_{i=0}^{3} i^n\right)$
- $n^2 \in \_\_\_\_ \left(\sum_{i=0}^{n} i\right)$
- $\sum_{i=0}^{3} i^n \in \_\_\_\_ (n^2)$
- $\left(\frac{8}{7}\right)^n \in \_\_\_\_ (n^2)$
- $\log_2(n) \in \_\_\_\_ (n)$

**b)** Beweisen oder widerlegen Sie:

$$f(n) \cdot g(n) \in \mathcal{O}\left(f(n)^{g(n)}\right)$$

## Aufgabe 2 (Rekursionsgleichungen): (10 + 8 + 8 = 26 Punkte)

**a)** Geben Sie für das Programm

```
int berechne(int n) {
    if (n <= 1)
        return 5000;

    int value = func(n);

    int k = 5;
    while (k >= 1) {
        value = value + value * berechne(n/3);
        k = k - 1;
    }

    return value;
}

int func(int n) {
    int res = 0;

    while (n > 0) {
        int m = n;
        while (m > 0) {
            res = res + m;
            m = m - 1;
        }
        n = n - 1;
    }

    return res;
}
```

eine Rekursionsgleichung für die **asymptotische** Laufzeit des Aufrufes `berechne(n)` in Abhängigkeit von `n` an. Die elementaren, also die für die asymptotische Laufzeit relevanten, Operationen sind alle arithmetischen Operationen sowie Vergleiche. Sie brauchen die Basisfälle der Rekursionsgleichung *nicht* anzugeben.

**b)** Bestimmen Sie für die Rekursionsgleichung

$$T(n) = 8 \cdot T\left(\frac{n}{2}\right) + n^3 + 4 \cdot n + \frac{1}{n}$$

die Komplexitätsklasse $\Theta$ mit Hilfe des Master-Theorems. Begründen Sie Ihre Antwort.

**c)** Sei $T(n)$ rekursiv wie folgt definiert:

$$T(n) \;=\; \begin{cases} 1, & \text{falls } n = 0 \\ 2 \cdot T(n-1), & \text{andernfalls.} \end{cases}$$

Beweisen Sie die folgende Aussage mittels vollständiger Induktion:

$$T(n) \in \Theta(2^n)$$

## Aufgabe 3 (Sortieren): (4 + 5 + 3 = 12 Punkte)

**a)** Sortieren Sie das folgende Array mithilfe von Insertionsort. Geben Sie dazu das Array nach jeder Iteration der äußeren Schleife an. Die vorgegebene Anzahl an Zeilen muss nicht mit der benötigten Anzahl an Zeilen übereinstimmen.

| 4 | 3 | 8 | 7 | 2 |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**b)** Sortieren Sie das folgende Array mithilfe von Quicksort. Geben Sie dazu das Array nach jeder Partition-Operation an und markieren Sie das jeweils verwendete Pivot-Element. Die vorgegebene Anzahl an Zeilen muss nicht mit der benötigten Anzahl an Zeilen übereinstimmen.

| 9 | 7 | 3 | 1 | 6 | 8 | 2 | 5 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

**c)** Wenden Sie die buildHeap Operation aus der Vorlesung (also den Anfang von Heapsort) auf das folgende Array an, um darauf die Max-Heap-Eigenschaft herzustellen. Geben Sie dazu das Array nach jeder Swap-Operation an. Die vorgegebene Anzahl an Zeilen muss nicht mit der benötigten Anzahl an Zeilen übereinstimmen.

| 4 | 7 | 8 | 3 | 2 | 9 |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

## Aufgabe 4 (Hashing): (3 + 3 = 6 Punkte)

**a)** Fügen Sie die folgenden Werte in das unten stehende Array a der Länge 10 unter Verwendung der *Divisionsmethode* mit *linearer Sondierung* ein:

3, 15, 13, 24, 23, 12.

| a[0] | a[1] | a[2] | a[3] | a[4] | a[5] | a[6] | a[7] | a[8] | a[9] |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

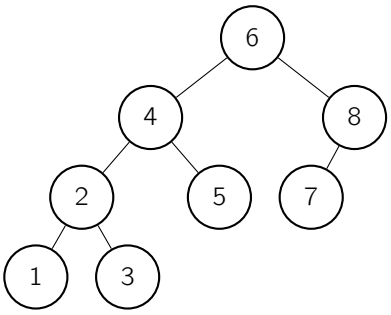
**b)** Fügen Sie die folgenden Werte in das unten stehende Array a der Länge 10 unter Verwendung der *Divisionsmethode* mit *quadratischer Sondierung* ($c_1 = 0.0$, $c_2 = 1.0$) ein:

7, 28, 17, 10, 20, 27.

| a[0] | a[1] | a[2] | a[3] | a[4] | a[5] | a[6] | a[7] | a[8] | a[9] |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

## Aufgabe 5 (Bäume): (4 + 7 + 6 = 17 Punkte)

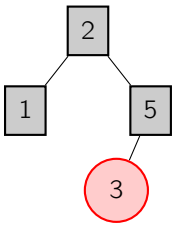
**a)** Löschen Sie den Wert 8 aus dem folgenden *AVL-Baum* und geben Sie die entstehenden Bäume nach jeder *Löschoperation* sowie jeder *Rotation* an. Markieren Sie außerdem zu jeder Rotation, welcher Knoten in welche Richtung rotiert wird:



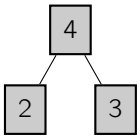
**b)** Fügen Sie den Wert 4 in den folgenden *Rot-Schwarz-Baum* ein und geben Sie die entstehenden Bäume nach

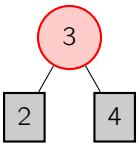
- jeder *Einfügeoperation*,
- jeder *Rotation* sowie
- jeder *Umfärbung* an.

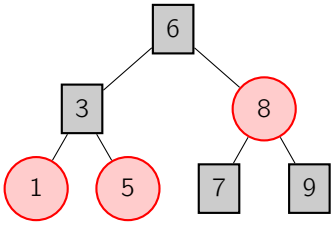
Markieren Sie außerdem zu jeder Rotation, welcher Knoten in welche Richtung rotiert wird. Mehrere Umfärbungen können Sie in einem Schritt zusammenfassen. Beachten Sie, dass rote Knoten rund und schwarze Knoten eckig dargestellt werden.

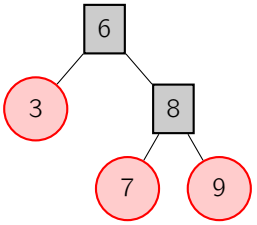


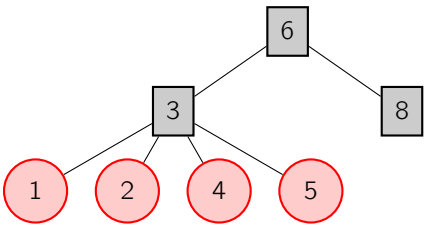
**c)** Geben Sie zu den folgenden Bäumen an, ob es sich dabei jeweils um einen gültigen Rot-Schwarz-Baum handelt. Falls dies nicht der Fall sein sollte, geben Sie mindestens eine Eigenschaft eines Rot-Schwarz-Baums an, die der jeweilige Baum verletzt. Beachten Sie, dass rote Knoten rund und schwarze Knoten eckig dargestellt werden.

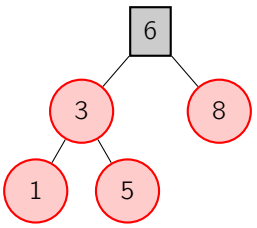


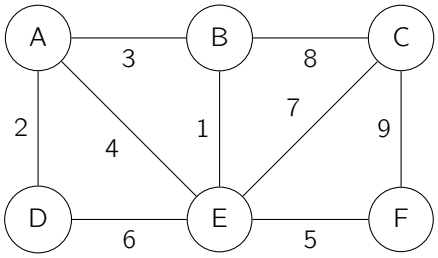



## Aufgabe 6 (Graphen): (6 + 8 + 3 + 6 = 23 Punkte)

**a)** Führen Sie Prim's Algorithmus auf dem folgenden Graphen aus.




Der Startknoten hat hierbei den Schlüssel A. Geben Sie dazu *vor* jedem Durchlauf der äußeren Schleife an,

a) welche Kosten die Randknoten haben (d. h. für jeden Knoten `v` in `pq` die Priorität von `v`, wobei $\infty$ angibt, dass der entsprechende Knoten noch nicht zum Randbereich gehört)

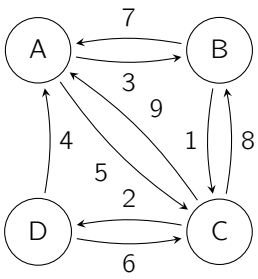
b) und welchen Knoten `pq.getMin()` wählt, indem Sie den Kosten-Wert des gewählten Randknoten in der Tabelle unterstreichen (wie es in der ersten Zeile bereits vorgegeben ist).

Geben Sie zudem den vom Algorithmus bestimmten minimalen Spannbaum an.

| #Iteration | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | $\underline{0}$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |
| 6 | | | | | | |

Minimaler Spannbaum:

**b)** Betrachten Sie den folgenden Graphen:




Führen Sie den *Algorithmus von Floyd* auf diesem Graphen aus. Geben Sie dazu nach jedem Durchlauf der äußeren Schleife die aktuellen Entfernungen in einer Tabelle an. Die erste Tabelle enthält bereits die Adjazenzmatrix nach Bildung der reflexiven Hülle. Der Eintrag in der Zeile $i$ und Spalte $j$ ist also $\infty$, falls es keine Kante vom Knoten der Zeile $i$ zu dem Knoten der Spalte $j$ gibt, und sonst das Gewicht dieser Kante. Beachten Sie, dass in der reflexiven Hülle jeder Knoten eine Kante mit Gewicht 0 zu sich selbst hat.

| (1) | A | B | C | D |
|---|---|---|---|---|
| A | 0 | 3 | 5 | $\infty$ |
| B | 7 | 0 | 1 | $\infty$ |
| C | 9 | 8 | 0 | 2 |
| D | 4 | $\infty$ | 6 | 0 |

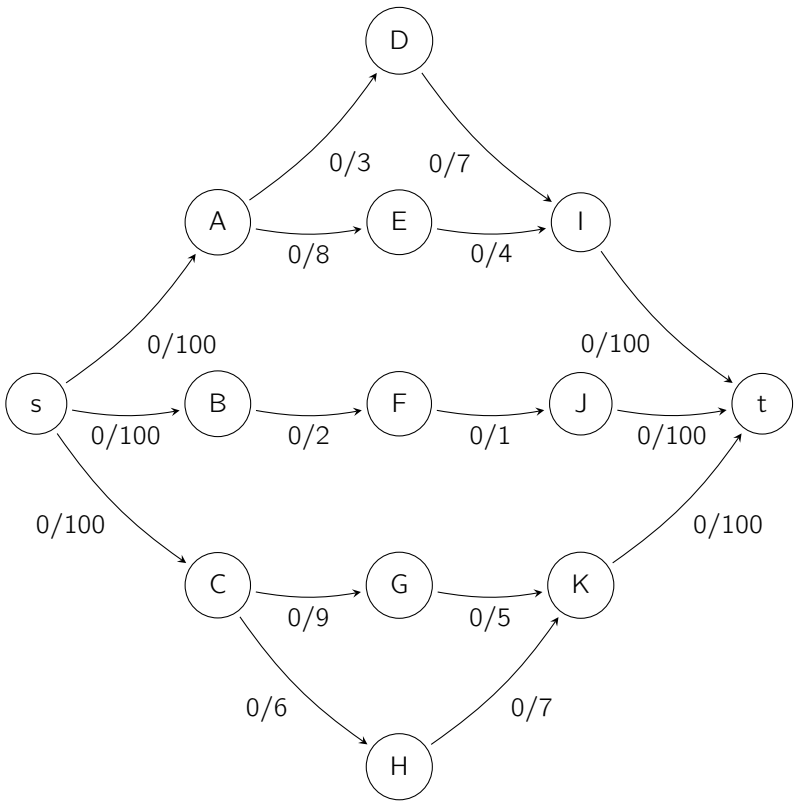
| (2) | A | B | C | D |
|---|---|---|---|---|
| A | | | | |
| B | | | | |
| C | | | | |
| D | | | | |

| (3) | A | B | C | D |
|---|---|---|---|---|
| A | | | | |
| B | | | | |
| C | | | | |
| D | | | | |

| (4) | A | B | C | D |
|---|---|---|---|---|
| A | | | | |
| B | | | | |
| C | | | | |
| D | | | | |

| (5) | A | B | C | D |
|---|---|---|---|---|
| A | | | | |
| B | | | | |
| C | | | | |
| D | | | | |

**c)** Betrachten Sie das folgende Flussnetzwerk mit Quelle s und Senke t:




Geben Sie einen minimalen Schnitt sowie den Wert des maximalen Flusses in diesem Flussnetzwerk an.

**d)** Gegeben ist eine endliche Menge von Servern $S$ und eine Funktion $b\colon S \times S \to \mathbb{N}$. Für zwei Server $s_1, s_2 \in S$ gibt der Funktionswert $b(s_1, s_2)$ die Bandbreite an, mit der Daten vom Server $s_1$ *direkt* (d. h. ohne Weiterleitung über einen anderen Server) zum Server $s_2$ gesendet werden können (es gilt hierbei nicht zwingend $b(s_1, s_2) = b(s_2, s_1)$).

Sollen nun Daten von $s_1$ nach $s_k$ gesendet werden, so kann es jedoch sein, dass die maximale Bandbreite entlang eines Pfades von Servern $s_1, \ldots, s_k$ höher ist als die Bandbreite $b(s_1, s_k)$ der direkten Verbindung. Da Bandbreiten Flaschenhälse darstellen, ist die maximale Bandbreite entlang des Pfades $s_1, \ldots, s_k$ durch das Minimum

$$\min\left\{b(s_1, s_2),\ b(s_2, s_3),\ \ldots,\ b(s_{k-1}, s_k)\right\}$$

gegeben.

Entwerfen Sie (in Stichpunkten) einen Algorithmus, der zwischen allen geordneten Paaren von Servern die maximale Bandbreite ermittelt. Der Algorithmus soll dabei insgesamt in Zeit $\Theta\left(|S|^3\right)$ ausgeführt werden können, d. h. nach $\Theta\left(|S|^3\right)$ Schritten sollen zwischen *allen* Paaren die maximalen Bandbreiten errechnet worden sein. Der Algorithmus soll eine Abwandlung eines Ihnen aus der Vorlesung bekannten Algorithmus zur Berechnung *kürzester Pfade* sein. Geben Sie insbesondere an, welchen Algorithmus Sie abwandeln. Begründen Sie kurz, warum ihr Algorithmus das vorliegende Problem löst.

## Aufgabe 7 (Dynamische Programmierung): (7 + 8 = 15 Punkte)

**a)** Gegeben sei ein Rucksack mit *maximaler Tragkraft* 7 sowie 4 *Gegenstände*. Der $i$-te Gegenstand soll hierbei ein Gewicht von $w_i$ und einen Wert von $c_i$ haben. Bestimmen Sie mit Hilfe des in der Vorlesung vorgestellten Algorithmus zum Lösen des Rucksackproblems mit dynamischer Programmierung den *maximalen Gesamtwert* der Gegenstände, die der Rucksack tragen kann (das Gesamtgewicht der mitgeführten Gegenstände übersteigt nicht die Tragkraft des Rucksacks). Die *Gewichte* seien dabei $w_1 = 4$, $w_2 = 5$, $w_3 = 3$ und $w_4 = 6$ und die *Werte* $c_1 = 3$, $c_2 = 4$, $c_3 = 2$ und $c_4 = 5$. Geben Sie zudem die vom Algorithmus bestimmte Tabelle C und die *mitzunehmenden Gegenstände* an.

Zur Erinnerung: Die Rekursionsgleichung für das Rucksackproblem lautet:

$$C[i,j] = \begin{cases} 0 & \text{für } i = 0, j \geq 0 \\ -\infty & \text{für } j < 0 \\ max(C[i-1,j], c_i + C[i-1, j-w_i]) & \text{sonst} \end{cases}$$

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | |
| 1 | | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |

Maximaler Wert:

Mitzunehmende Gegenstände:

**b)** Um weiterhin hochqualitative Vorlesungsvideos anzubieten, will die Video AG Werbespots in die Vorlesungsvideos einbauen. Um die Aufmerksamkeit der Zuschauer nicht zu oft zu unterbrechen, ist entschieden worden, dass es genau zwei Werbeunterbrechungen in einem Vorlesungsvideo geben soll, deren Länge jeweils maximal $k$ Zeiteinheiten beträgt. Der Video AG liegen Angebote für Werbespots aus der Menge $W = \{w_1, \ldots, w_n\}$ vor. Die Länge des Werbespots $w_i$ ist durch $\ell_i$ Zeiteinheiten gegeben. Da die beiden Werbeblöcke als unterschiedlich wirksam erachtet werden, wird ein Preis $p_{i,1}$ erzielt, wenn der Werbespot $w_i$ in der ersten Werbepause gesendet wird, und ein Preis $p_{i,2}$, wenn der Werbespot in der zweiten Werbepause gesendet wird. Jeder Werbespot darf aber insgesamt höchstens einmal während eines Vorlesungsvideos gesendet werden. Außerdem dürfen Werbespots nur in voller Länge gesendet werden (eine teilweise Sendung ist also nicht erlaubt).

Helfen Sie der Video AG, den Gewinn durch die Werbesendungen zu maximieren.

Gehen Sie dazu wie folgt vor:

(i) Stellen Sie eine Rekursionsgleichung $G(i, t_1, t_2)$ auf, die angibt, welcher Gewinn maximal erzielt werden kann, wenn die noch verfügbaren Zeiten der beiden Werbeblöcke $t_1$ bzw. $t_2$ Zeiteinheiten betragen und nur die Werbespots $W' = \{w_1, \ldots, w_i\}$ zur Verfügung stehen.

(ii) Geben Sie an, für welche Werte $i, t_1, t_2$ der Wert von $G(i, t_1, t_2)$ das gesuchte Ergebnis (also die maximalen Werbeeinnahmen unter den oben beschriebenen Randbedingungen) liefert. Anders formuliert: Welcher "Aufruf" der Rekursionsgleichung liefert das gewünschte Ergebnis?