Prof. Dr. Ir. Joost-Pieter Katoen

Christian Dehnert, Jonathan Heinen, Thomas Ströder, Sabrina von Styp

Präsenzübung Datenstrukturen und Algorithmen SoSe 2012

Vorname: ______________________________

Nachname: ______________________________

Matrikelnummer: ______________________________

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

- ○ Informatik Bachelor
- ○ Informatik Lehramt
- ○ Mathematik Bachelor
- ○ Computational Engineering Science
- ○ Sonstiges: ______________________

	Anzahl Punkte	Erreichte Punkte
Aufgabe 1	**10**	
Aufgabe 2	**10**	
Aufgabe 3	**10**	
Aufgabe 4	**10**	
Aufgabe 5	**10**	
Summe	**50**	

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 (benutzen Sie auch die Rückseiten).
- 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 Präsenzübung mit **0 Punkten** bewertet.
- Geben Sie am Ende der Präsenzübung **alle Blätter zusammen mit den Aufgabenblättern ab**.
- Gehen Sie bei der Codeanalysen davon aus, dass sämtliche Instruktionen wie arithmetische Operationen (+,-.*,/), Vergleiche usw. in konstanter Zeit $\mathcal{O}(1)$ ausgeführt werden.

Name: **Matrikelnummer:**

Aufgabe 1 ($\mathcal{O}$-Notation): (3 + 3 + 4 = 10 Punkte)

Beweisen oder widerlegen Sie die folgenden Aussagen:

a) $\log_2 n \in \Omega(\frac{1}{n})$

b) $2^n \in \Theta(3^n)$

c) $\sum_{i=0}^{n} i! \in \Theta(n!)$

Aufgabe 2 (Sortieren): (3 + 2 + 5 = 10 Punkte)

a) Sortieren Sie das folgende Array mittels des Mergesort-Algorithmus.

7	13	3	2	9	5	4	8

Geben Sie nach jeder Merge-Operation den jeweils verschmolzenen Arraybereich an.

b) Gegeben sei ein Array E, das Datenobjekte enthält, welche je zwei Schlüssel K1 und K2 besitzen. Anna sortiert das Array E mittels eines Sortieralgorithmus S zuerst nach den Schlüsseln K1. Anschließend sortiert sie diejenigen Arraybereiche, bei denen die Werte des Schlüssels K1 gleich sind, mit demselben Algorithmus S nach den Werten des zweiten Schlüssels K2. Boris sortiert das Array E mittels des Algorithmus S zuerst nach den Schlüsseln K2. Anschließend sortiert er das gesamte resultierende Array mit dem Algorithmus S nach den Schlüsseln K1.

Unter welchen Bedingungen erhalten Anna und Boris für ein beliebiges Eingabearray E immer das gleiche Ergebnis? Begründen Sie Ihre Antwort!

c) Betrachten Sie den folgenden Sortieralgorithmus `gnome`.

```
void gnome(int E[]) {
    int i = 0;
    while (i < E.length) {
        if (i == 0 || E[i-1] <= E[i]) {
            i++;
        } else {
            swap(E,i,i-1);
            i--;
        }
    }
}

void swap(int E[], int i1, int i2) {
    int store = E[i1];
    E[i1] = E[i2];
    E[i2] = store;
}
```

Bestimmen Sie die Worst-Case Laufzeit (Θ) dieses Algorithmus in Abhängigkeit der Arraylänge n und geben Sie an, ob dieser Algorithmus ein stabiler Sortieralgorithmus ist. Begründen Sie Ihre Antwort!

Name: **Matrikelnummer:**

Aufgabe 3 (Datenstruktur): (2 + 4 + 4 = 10 Punkte)

Zum effizienten Speichern einer Matrix M mit n Zeilen, m Spalten und z Nicht-Nulleinträgen verwaltet das sogenannte CRS-Format drei Arrays wie folgt:

- Das Array $values$ speichert ausschließlich die Nicht-Nulleinträge von M in fortlaufender Reihenfolge, wobei diese Werte zeilenweise von links nach rechts abgelesen werden.
- Das Array $columns$ speichert zu jedem Nicht-Nulleintrag in $values$ die zugehörige Spalte, die dieser Wert in der Matrix einnimmt.
- Das Array $rows$ speichert für jede Zeile i von M den Index des ersten Elements der Zeile im Array $values$. Um anzuzeigen, wo die Werte der letzten Zeile in $values$ enden, enthält es am Schluss noch ein zusätzliches Sentinel-Element mit dem Wert $length(values)$.

Betrachten wir nun beispielsweise die Darstellung der Matrix

$$\begin{pmatrix} 10 & 0 & 2 & 0 & 0 \\ 0 & 5 & 0 & 13 & 0 \\ 0 & 0 & 18 & 0 & 0 \\ 4 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{pmatrix}$$

Die entsprechenden Arrays des CRS-Formats sind gegeben durch

- $values = \begin{bmatrix} 10 & 2 & 5 & 13 & 18 & 4 & 1 & 3 \end{bmatrix}$
- $columns = \begin{bmatrix} 1 & 3 & 2 & 4 & 3 & 1 & 4 & 5 \end{bmatrix}$
- $rows = \begin{bmatrix} 0 & 2 & 4 & 5 & 7 & 8 \end{bmatrix}$

Sei M nun eine beliebige Matrix mit n Zeilen, m Spalten und z Nicht-Nulleinträgen.

a) Geben Sie die exakte Anzahl D der Dateneintrage, die das CRS-Format für M speichern muss, in Abhängigkeit der Parameter n, m und z an. Als Dateneinträge werden hierbei alle Inhalte der Arrays $values$, $columns$ und $rows$ aufgefasst.

b) Das folgende Codefragment setzt einen Wert a in der i-ten Zeile und j-ten Spalte in eine CRS-Matrix ein ($1 \leq i \leq n$ und $1 \leq j \leq m$). Es wird vereinfachend davon ausgegangen, dass hinter das Ende eines Arrays sicher geschrieben werden kann und sich dadurch die Länge des Arrays entsprechend vergrößert.

```
public static void insert(CRSMatrix M, int a, int i, int j) {
    // Bestimme den Anfangsindex der Zeilen i und i+1
    int rowIndex = M.rows[i - 1];
    int nextRowIndex = M.rows[i];

    // Finde Position in Zeile i an der eingefuegt werden muss
    int pos = rowIndex;
    while (pos < nextRowIndex && j > M.columns[pos]) {
        pos++;
    }

    // Eintrag existiert schon
    if (pos < nextRowIndex && j == M.columns[pos]) {
        M.values[pos] = a; // ueberschreibe nur den Wert
    } else { // Eintrag existierte nicht
        // verschiebe entsprechende Arrayelemente
        for (int k = M.values.length - 1; k >= pos; k--) {
            M.values[k + 1] = M.values[k];
            M.columns[k + 1] = M.columns[k];
        }

        // setze Wert und Spalte
        M.values[pos] = a;
        M.columns[pos] = j;

        // die auf i folgenden Zeilen beginnen nun einen Index spaeter
        for (int k = i; k < M.rows.length; k++) {
            M.rows[k]++;
        }
    }
}
```

Geben Sie die Laufzeitkomplexität von $insert(M, a, i, j)$ in Abhängigkeit der Zeilen n, Spalten m und Nicht-Nulleinträge z für den Best-Case als auch den Worst-Case an. Begründen Sie ihre Antwort.

c) Es soll nun ein Algorithmus entworfen werden, der für eine natürliche Zahl k das erste Vorkommen dieses Wertes bezüglich der Reihenfolge in $values$ in M bestimmt und dann ausgibt, in welcher Zeile und in welcher Spalte von M dieser Wert steht. Kommt k nicht in M vor, so soll nichts ausgegeben werden.

Beschreiben Sie **in Stichpunkten** ein möglichst effizientes Verfahren, welches die Aufgabe erfüllt, und geben Sie die resultierende Worst-Case-Laufzeitkomplexität ihres Verfahrens an. Orientieren Sie sich hierbei an bekannten Algorithmen aus der Vorlesung.

Aufgabe 4 (Rekursionsgleichungen): (4 + 2 + 4 = 10 Punkte)

a) Geben Sie für das folgende Programm eine Rekursionsgleichung für die asymptotische Laufzeit des Aufrufes `berechne(n)` an.

```
int berechne(int n){
    if(n <= 0)
        return 3;

    int value = 4;
    for(int i = 0; i < n*n; i++){
        value += pow(i);
    }

    value += 3 * berechne(n/2) + 4 * berechne(n/4) + 5
    return value;
}

int pow(n){
    int value = n;
    for(int i = 0; i < n; i++)
        value = value *n;
    return value;
}
```

b) Bestimmen Sie für die folgende Rekursionsgleichung die Komplexitätsklasse Θ mit Hilfe des Mastertheorems. Begründen Sie Ihre Antwort.

$$T(n) = 4T(n/2) + n \cdot \log_2 n + 4n + 3$$

c) Skizzieren Sie den Rekursionsbaum zu der folgenden Rekursionsgleichung. Lesen Sie die Komplexitätsklasse Θ der Gleichung ab. Die abgelesene Komplexitätsklasse **muss nicht bewiesen**, **Summen nicht aufgelöst** werden. Eine Angabe der Laufzeit der Art $\Theta(\sum \ldots)$ ist zulässig.

$$T(n) = 3 \cdot T(n-2) + 2n^2, \quad T(0) = T(1) = 1$$

Aufgabe 5 (Bäume): (2 + 1 + 3 + 1 + 3 = 10 Punkte)

Ein höhenbalancierter Baum ist ein Binärbaum, bei dem sich für jeden Knoten die Höhe seiner Teilbäume höchstens um 1 unterscheiden darf.

a) Zeichnen Sie einen höhenbalancierten Baum der Höhe 3 mit **minimaler Anzahl** an Knoten. Geben Sie für jeden Knoten die Höhe des Teilbaumes an, der an diesem Knoten beginnt.
Hinweis: Der zu zeichnende Baum hat 7 Knoten.

b) 1. Stellen Sie eine Rekursionsgleichung $E(h)$ auf, mit der sich die **minimale Anzahl** von Knoten in einem höhenbalancierten Baum der Höhe h bestimmen lässt. *Geben Sie auch die nötigen Basisfälle an.*

2. Beweisen Sie, dass $\frac{1}{2} \cdot (\frac{3}{2})^h + \frac{1}{2}$ eine *untere Schranke* für die Anzahl der Knoten ist.

Name: **Matrikelnummer:**

c) Geben Sie die minimale Höhe $h(n)$ eines höhenbalancierten Baumes mit n Knoten an. Begründen Sie Ihre Antwort kurz. Ein formaler Beweis ist nicht nötig.

d) Zeigen Sie, dass für die Höhe $h(n)$ eines höhenbalancierten Baumes mit n Knoten gilt, dass $h(n) \in \mathcal{O}(\log_2 n)$. Zum Lösen dieser Aufgabe dürfen Sie die untere Schranke $\frac{1}{2} \cdot (\frac{3}{2})^h + \frac{1}{2}$ aus Aufgabe **5 b)** nutzen.