Westfälische Wilhelms-Universität Münster
Übungen zur Vorlesung „Datenstrukturen und Algorithmen“ im SoSe 2017

Prof. Dr. Klaus Hinrichs
Aaron Scherzinger

Blatt 8

Abgabe im Learnweb bis zum 29.06.2017 um 10 Uhr

Aufgabe 26: (2+8+8=18 Punkte) Sei M eine antisymmetrische $n \times n$-Matrix, d.h., für alle $i, j \in \{1, \ldots, n\}$ gilt: $M_{ij} = -M_{ji}$. Jeder Eintrag sei eine Zahl vom Typ `double`.

(a) Welche Werte haben die Elemente auf der Hauptdiagonalen, d.h., die Werte M_{ii} für $1 \leq i \leq n$.

(b) Wie kann M in einem Array `double[] a` gespeichert werden, sodass die Länge c des Arrays minimal (insbesondere kleiner als $n \cdot n$) ist? Wie groß ist c in Abhängigkeit von n?

(c) Implementieren Sie Ihre in (b) enwickelte Repräsentation durch eine Klasse `MyMatrix`. Schreiben Sie für diese Klasse insbesondere eine `JAVA`-Methode

```
public double get(int i, int j),
```

sodass der Aufruf von `get(i,j)` den Wert M_{ij} zurückgibt. Betrachten Sie auch den Fall, dass die Parameter i bzw. j außerhalb des Bereiches $[1, \ldots, n]$ liegen können. Verwenden Sie dabei Exceptions.

Aufgabe 27: (5+10=15 Punkte) In der Vorlesung wurde der abstrakte Datentyp `DICT` definiert, auf dem die vier Operationen `create`, `insert`, `member` und `delete` definiert sind. Ein Dictionary unterstützt allerdings gewöhnlich weitere Operationen basierend auf einer totalen Ordnung $\leq$ auf dem Wertebereich der verwalteten Elemente. So sind etwa zwei Operationen `succ` und `pred` denkbar, die zu einem gegebenen Element `x` aus dem Dictionary den Nachfolger (*successor*) und den Vorgänger (*predecessor*) zurückliefern. Der Nachfolger ist als das kleinste derjenigen Elemente aus dem Dictionary definiert, die größer als `x` sind, während der Vorgänger das größte derjenigen im Dictionary enthaltenen Elemente darstellt, die kleiner als `x` sind. Sollte kein Nachfolger oder kein Vorgänger zu einem Element `x` existieren, kann ein spezielles Element `largestElem` oder `smallestElem` zurückgegeben werden.
Erweitern Sie den ADT `DICT` um die folgenden Operationen.

(a) Fügen Sie zwei Operationen `min` und `max` hinzu, die jeweils das kleinste und größte im Dictionary gespeicherte Element zurückliefern. Sie können sich dabei auf zwei Operationen $\texttt{min}_{\texttt{ELEM}}$ und $\texttt{max}_{\texttt{ELEM}}$ abstützen, die auf dem Datentyp `ELEM` definiert sein müssen und von zwei übergebenen Elementen das jeweils kleinere und größere bestimmen.

(b) Erweitern Sei den ADT `DICT` um die beiden oben beschriebenen Operationen `succ` und `pred`. Sie können dazu eine Operation $\texttt{isequal}_{\texttt{ELEM}}$ verwenden, die zwei übergebene Elemente auf Gleichheit testet und als Resultat ein Element vom Typ `BOOL` erzeugt.

Aufgabe 28: (8 Punkte) Schreiben Sie eine Methode

```
public int[][] multipliziereBandmatrizen(int[][] a1, int[][] a2, int n, int b)
```

zur Multiplikation zweier Bandmatrizen `a1` und `a2` der Größe $n \times n$, die ganze Zahlen enthalten. Die Repräsentation der Bandmatrizen soll so realisiert werden, wie es auf den Vorlesungsfolien (Kap. 7, Folien 9-10) beschrieben wird (n groß, b klein). Die Bandmatrizen sollen beide die Breite $2 \cdot b + 1$ besitzen. Dokumentieren Sie Ihre Methode gemäß `JAVADOC`-und testen Sie Ihre Implementation. Bestimmen Sie zudem die asymptotische Laufzeit Ihres Algorithmus in Abhängigkeit von n und b.

Hinweis: Arbeiten Sie auch intern in Ihrer Methode mit der Repräsentation der Bandmatrizen aus der Vorlesung. Konvertieren Sie zu keinem Zeitpunkt die Matrizen in eine andere Darstellung.

Aufgabe 29: (4+8=12 Punkte) Bearbeiten Sie die folgenden Aufgaben.

(a) Beschreiben Sie umgangssprachlich oder in Pseudo-Code einen Algorithmus, der für ein gegebenes Array von n paarweise verschiedenen ganzen Zahlen in $\Theta(n \log n)$ Zeit alle (geordneten) Paare (a, b) von Elementen dieses Arrays bestimmt, deren Summe einer ebenfalls übergebenen Zahl s entspricht.

(b) Implementieren Sie Ihren Algorithmus in `JAVA`, indem Sie eine Methode mit folgender Signatur schreiben:

```
public int[][] computePairsForSum(int[] values, int sum)
```

Kommentieren Sie Ihre Methode gemäß `JAVADOC` und entwickeln Sie geeignete Testfälle.

Bonusaufgabe(*):

Aufgabe 30: (8+4+4+4=20 Punkte) Betrachten Sie die asymptotischen Laufzeitkomplexitäten der unten angegebenen Funktionen.

(a) Geben Sie zu jeder der folgenden Funktionen $f_i : \mathbb{N} \to \mathbb{R}^+$, $1 \leq i \leq 4$, eine möglichst einfache Funktion $g_i : \mathbb{N} \to \mathbb{R}^+$, $1 \leq i \leq 4$, an, so dass gilt: $f_i \in \mathcal{O}(g_i)$, $1 \leq i \leq 4$ (Hierbei ist kein Beweis gefordert!). Die Funktion g_i soll eine scharfe obere Schranke darstellen.

$f_1 : \mathbb{N} \to \mathbb{R}^+,\ n \mapsto n^3 \cdot \log_2 n + 0.03 \cdot n^4 + 2 \cdot n$ $\quad$ $g_1 : \mathbb{N} \to \mathbb{R}^+,\ n \mapsto$

$f_2 : \mathbb{N} \to \mathbb{R}^+,\ n \mapsto n \cdot \log_2 n + n \cdot \log_2^2 n + 1234 \cdot n$ $\quad$ $g_2 : \mathbb{N} \to \mathbb{R}^+,\ n \mapsto$

$f_3 : \mathbb{N} \to \mathbb{R}^+,\ n \mapsto 5 \cdot n \cdot \log_2(\log_2 n) + n^2$ $\quad$ $g_3 : \mathbb{N} \to \mathbb{R}^+,\ n \mapsto$

$f_4 : \mathbb{N} \to \mathbb{R}^+,\ n \mapsto 2 \cdot \log_2^2 n + 34 \cdot \log_2(\log_2 n) + 0.3 \cdot \log_2 n$ $\quad$ $g_4 : \mathbb{N} \to \mathbb{R}^+,\ n \mapsto$

(b) Betrachten Sie die Funktionen $f : \mathbb{N} \to \mathbb{R}^+,\ n \mapsto \log_2(n^n)$ und $g : \mathbb{N} \to \mathbb{R}^+,\ n \mapsto n^2 \log_2 n$. Beweisen Sie formal, dass gilt: $f \in \mathcal{O}(g)$.

(c) Betrachten Sie die Funktionen $f : \mathbb{N} \to \mathbb{R}^+,\ n \mapsto n \log_4 n$ und $g : \mathbb{N} \to \mathbb{R}^+,\ n \mapsto n \log_2 n$. Beweisen Sie formal, dass gilt: $f \in \Omega(g)$.

(d) Betrachten Sie die folgende Funktion:

$$f : \mathbb{N} \to \mathbb{R}^+,\ n \mapsto \begin{cases} n! & \text{für } 1 \leq n \leq 17 \\ 2^{2^n} & \text{für } 18 \leq n \leq 42 \\ \log_2 n & \text{für } n \geq 43 \end{cases}$$

Geben Sie eine möglichst genaue obere Schranke für f an, d.h. eine möglichst langsam wachsende Funktion $g : \mathbb{N} \to \mathbb{R}^+$, für die gilt: $f \in \mathcal{O}(g)$. Beweisen Sie formal, dass gilt: $f \in \mathcal{O}(g)$.

(*)**Hinweis**: Bei dieser Aufgabe handelt es sich um eine Bonusaufgabe, d.h. Sie können mit dieser Aufgabe zusätzliche Punkte in der Übung erwerben, es ist aber möglich, ohne diese Aufgabe 100% der Übungspunkte zu erreichen. Es handelt sich um eine Wiederholung bereits bekannten Stoffs. Wir empfehlen allen Teilnehmerinnen und Teilnehmern die Bearbeitung der Aufgabe.