Institut für Computergraphik
E186/1 Informatik

Abteilung für Algorithmen und Datenstrukturen

TECHNISCHE UNIVERSITÄT WIEN

Univ.-Prof. Dr. Petra Mutzel
Dipl.-Math. Katja Bühler
Dipl.-Inform. René Weiskircher
Dipl.-Inform. Thomas Ziegler

Sommersemester 2000

Prüfung zur Vorlesung Algorithmen und Datenstrukturen 1 30. Juni 2000

1. Machen Sie bitte die folgenden Angaben in deutlicher Blockschrift:

   Name: _____ Vorname: _____

   Matrikelnummer: _____ Studienkennzahl: _____

2. Legen Sie während der Vorlesungsprüfung Ihren Studentenausweis vor sich auf das Pult.
3. Schreiben Sie die Lösungen direkt auf das jeweilige Aufgabenblatt. Wenn Ihnen das Papier ausgeht, bitten Sie die Aufsicht um Nachschub. Es ist nicht erlaubt eigenes Papier zu verwenden!
4. Denken Sie daran, dass keinerlei Hilfsmittel erlaubt sind (weder Taschenrechner, irgendwelche Unterlagen, Handy's,...).

VOR DER ABGABE AUSZUFÜLLEN:

5. Geben Sie bitte die Anzahl der zusätzlich abgegebenen Blätter an: _____
6. Kreuzen Sie bitte die von Ihnen bearbeiteten Aufgaben in der ersten Zeile der Tabelle an:

Aufgabe	A 1	A 2	A 3	A 4	A 5	A 6	$\sum$	Note
bearbeitet							—	_____
maximale Punktzahl	10	6	12	9	7	6	50	_____
erreichte Punktzahl								

Viel Erfolg!

Aufgabe 1: O/Θ/Ω-Notation **10 Punkte**

a.) Kreuzen Sie **alle** richtigen Aussagen an. Beachten Sie, dass es durchaus mehrere richtige Antworten pro Zeile geben kann, die alle angekreuzt werden müssen, um volle Punktzahl zu erreichen!

f(n)	g(n)	$f(n) = \Omega(g(n))$	$f(n) = \Theta(g(n))$	$f(n) = O(g(n))$
$\sqrt{n}$	$750n$			
$2n^2$	$5n \log n$			
$n^4 + 26n^3 + 6n^2 + n + 8$	$100n^4$			
$n!$	7^n			

Bewertung: Für jedes richtige Kreuz gibt es einen Pluspunkt, für jedes falsche Kreuz einen Punkt Abzug, für fehlende Kreuze keine Punkte. Es können insgesamt nicht weniger als 0 Punkte erreicht werden. D.h. kreuzen Sie am Besten nur die Antworten an, bei denen Sie sich sicher sind.

b.) Sei

$$f(n) = \left\{ \begin{array}{ll} n & \text{für } n \leq 100 \\ 2n^2 - 3n & \text{für } n > 100 \end{array} \right\}$$

Ist $f(n) = \Omega(n^2)$? Beweisen oder widerlegen Sie diese Aussage.

Aufgabe 2: Heaps/Heapsort **6 Punkte**

a.) Was ist der wesentliche Unterschied zwischen einem Heap und einem binären Suchbaum.

b.) Sortieren Sie die Zahlenfolge

76	47	8	4	82	95

aufsteigend mit Hilfe von *Heapsort*. Stellen Sie dazu *jede* dabei entstehende Zwischenfolge in der Form eines binären Baumes dar.

Aufgabe 3: Binäre Suchbäume **12 Punkte**

a.) Geben Sie den Pseudocode für die Einfügefunktion `insert` für natürliche (unbalancierte) binäre Suchbäume an. Verwenden Sie hierzu die verallgemeinerte Listenstruktur, die über `x.key`, `x.left`, `x.right` und `x.father` ansprechbar ist.

b.) Gegeben sei eine *aufsteigend sortierte* Folge von n ganzen Zahlen, die in einem Array $A[1, ..., n]$ gespeichert sind. Diese Folge soll mit Hilfe der in a.) beschriebenen Funktion `insert` derart in einen binären Suchbaum eingetragen werden, dass dieser höchstens die Tiefe $\log_2 n$ besitzt. Geben Sie einen Algorithmus in Pseudocode an, der diese Aufgabe erfüllt. Beschreiben Sie auch die Idee Ihres Algorithmus.

Achten Sie darauf, dass der von Ihnen verwendete Pseudocode eindeutig ist. Verwenden Sie aussagekräftige Bezeichner und versehen Sie Ihren Pseudocode mit Kommentaren.

Aufgabe 4: Was macht der folgende Algorithmus? **9 Punkte**

Gegeben sei ein Array $A[1, ..., n]$ mit n ganzen Zahlen.

```
(0) Algorithmus mach_irgendwas (A[1, ..., n])
(1) /* Input: Folge von n Zahlen A[1, ..., n] */
(2) /* Output: ??? */
(3)     Für i = n − 1, ..., 1
(4)         Für j = 1, ..., i
(5)             Falls A[j] > A[j + 1]
(6)                 Vertausche(A[j], A[j + 1]);
```

a.) Was können Sie nach jeder Iteration der Schleife in Zeile (3) über das Element A[i] aussagen?

b.) Erklären Sie in eigenen Worten genau wie der Algorithmus funktioniert.

c.) Wie sieht der Output des Algorithmus aus?

d.) Geben Sie jeweils eine Eingabefolge A für den Worst-Case und den Best-Case bezüglich der Anzahl der Vertauschungen an.

e.) Berechnen Sie jeweils die Anzahl der Vertauschungen und Schlüsselvergleiche im Worst-Case und Best-Case.

f.) Geben Sie die Ergebnisse von e.) in Θ-Notation an.

Aufgabe 5: Abstrakte Datentypen **7 Punkte**

Der abstrakte Datentyp **Schlange** ist folgendermassen definiert:

Objekte:	Menge aller endlichen Folgen eines geg. Grundtyps T
Schreibweise:	Schlange mit Elementen $a_1, a_2, \ldots, a_n$: $Q = (a_1, a_2, \ldots, a_n)$ leere Schlange: $Q = ()$

Operationen:

- **Push(x,Q): Füge Element x in die Schlange ein**
  Falls $Q = ()$: vorher $Q = ()$, nachher $Q = (x)$.
  Falls $Q \neq ()$: vorher $Q = (a_1, a_2, \ldots, a_n)$, nachher $Q = (x, a_1, a_2, \ldots, a_n)$.
- **Pop(Q): Entferne letztes Element aus der Schlange**
  Falls $Q = ()$: undefiniert (Fehlermeldung).
  Falls $Q \neq ()$: vorher $Q = (a_1, a_2, \ldots, a_{n-1}, a_n)$, nachher $Q = (a_1, a_2, \ldots, a_{n-1})$.
- **Empty(Q): Teste, ob Schlange leer**
  Falls $Q = ()$: Rückgabe TRUE.
  Falls $Q \neq ()$: Rückgabe FALSE.

Geben Sie eine Implementierung des abstrakten Datentyps **Schlange** mit einer doppelt verketteten Liste mit Dummy-Element in Java–Code an. **Jede Operation sollte nur Zeit $O(1)$ benötigen.** Beachten Sie die dabei möglicherweise auftretenden Sonderfälle.

Benutzen Sie hierfür die auf der nächsten Seite angegebene Klasse.

Klassendefinition zur Aufgabe 5

```
class Listenelement {

    Object data;      // hier werden die Daten gespeichert
    Listenelement next, prev;      // Referenz auf Nachfolge-/Vorgängerknoten

    public Listenelement() {      // default-Konstruktor
       data = null;
       next = null; prev = null;
    }

    public Listenelement(Object x) {      // Konstruktor mit Datenzuweisung
       data = x;
       next = null; prev = null;
    }

}

public class Schlange {

    private Listenelement listbegin,      // Referenz auf den Listenbeginn (= Dummy-Element)
    listend;      // Referenz auf das letzte Element der Liste

    public Schlange() {      // Konstruktor; erzeugt Dummy-Element
       listbegin = new Listenelement();
       listend = listbegin;
    }

    public void push(Object x) {
       ...
    }

    public Object pop() {
       ...
    }

    public boolean empty() {
       ...
    }

}
```

Aufgabe 6: Hash-Verfahren **7 Punkte**

Sie sollen die Übungsteilnehmer der Vorlesung *Algorithmen und Datenstrukturen* anhand ihrer Matrikelnummer (=Schlüssel) in eine Hashtabelle einfügen. Es nehmen 700 Studierende an der Übung teil; die Hashtabelle hat Größe 1000. (Zur Erinnerung: Matrikelnummern sind 7-stellige Zahlen, z.B. 9922534, wobei die beiden führenden Ziffern das Immatrikulationsjahr angeben).

a.) Wir stellen Ihnen die beiden Hashfunktionen $h_1(k)$ und $h_2(k)$ zur Auswahl.

$$h_1(k) = \lfloor k/10000 \rfloor \quad (1)$$
$$h_2(k) = k \bmod 1000 \quad (2)$$

Welche von den beiden wählen Sie? Begründen Sie Ihre Auswahl.

b.) Weiterhin sollen die Praktikanten der Abteilung *Algorithmen und Datenstrukturen* in eine Hashtabelle T der Größe 5 eingetragen werden. Fügen Sie die folgenden Matrikelnummern mittels der Hashfunktion

$$h(k) = k \bmod 5$$

in T ein:

9924523, 9824624, 9936028, 9947282, 9732479, 9894628, 9958259

Verwenden Sie zur Kollisionsbehandlung die Methode der Verkettung der Überläufer. Stellen Sie die Hashtabelle nach jedem Einfügen dar.