Institut für Computergraphik und Algorithmen

Abteilung für Algorithmen und Datenstrukturen

TECHNISCHE UNIVERSITÄT WIEN

Prof. Petra Mutzel
Gunnar Klau
Ivana Ljubic

Wintersemester 2000/2001

Prüfung zur Vorlesung Algorithmen und Datenstrukturen 1 9. Oktober 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, Handys,...).

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	15	13	15	15	10	12	80	_____
erreichte Punktzahl								

Viel Erfolg!

Aufgabe 1: O-Notation/Sortierverfahren **15 Punkte**

a.) Geben Sie für die unten angegebenen Zahlenfolgen jeweils die Anzahl der Schlüsselvergleiche C sowie die Anzahl der Schlüsselbewegungen M der Sortierverfahren Insertion-Sort und Selection-Sort in Θ-Notation an. N steht für eine beliebig große gerade Zahl. Dabei sollen nur die echten Schlüsselvertauschungen gezählt werden.

Zahlenfolge	$C_{Ins-Sort}$	$M_{Ins-Sort}$	$C_{Sel-Sort}$	$M_{Sel-Sort}$
$\frac{N}{2}, \frac{N}{2}+1, \ldots, N, 1, 2, \ldots, \frac{N}{2}-1$				
$N, 1, 2, 3, \ldots, N-1$				
$N, 1, N-1, 2, N-2, 3, \ldots, N-\frac{N}{2}+1$				
$1, \frac{N}{2}+1, 2, \frac{N}{2}+2, \ldots, \frac{N}{2}, N$				
$2, 3, \ldots, N, 1$				

b.) Sei

$$f(n) = \frac{1}{10} n \log n$$

Ist $f(n) = \Omega(n)$? Was müssten Sie zeigen, um diese Aussage zu beweisen? Was müssten Sie zeigen, um diese Aussage zu widerlegen? Beweisen oder widerlegen Sie diese Aussage.

Aufgabe 2: Balancierte Binäre Suchbäume **13 Punkte**

a.) Konstruieren Sie den balancierten AVL–Baum, der sich aus dem leeren Baum ergibt, wenn man nacheinander die folgenden Elemente einfügt:

8, 5, 2, 15, 7, 10, 11, 6

Stellen Sie dabei jeden erzeugten Baum dar.

b.) Als nächstes löschen Sie das Element **5** und geben die erreichte Konfiguration an.

Aufgabe 3: Binäre Suchbäume **15 Punkte**

Gegeben sei ein Binärbaum B mit ganzzahligen Schlüsseln. Gegeben sei außerdem ein Schlüssel x. Gesucht ist in B der größte Schlüssel $\leq x$.

Geben Sie den Pseudocode für einen Algorithmus `Search` an, der diese Aufgabe in $O(h)$ Schritten löst, wenn h die Höhe von B ist. Dabei können Sie davon ausgehen, dass für jedes x ein größter im Binärbaum gespeicherter Schlüssel mit Wert $\leq x$ stets vorkommt, da im Binärbaum ein unechter Schlüssel mit Wert $-\infty$ gespeichert ist.

Verwenden Sie hierzu die verallgemeinerte Listenstruktur, die über `p.key`, `p.left`, `p.right` und `p.father` ansprechbar ist. Achten Sie darauf, dass der von Ihnen verwendete Pseudocode eindeutig ist. Verwenden Sie aussagekräftige Bezeichner und versehen Sie Ihren Pseudocode mit Kommentaren.

In Ihrem Pseudocode dürfen Sie folgende Funktionen benutzen:

`Minimum (p)`	gibt das kleinste Element des Teilbaums mit Wurzel `p` zurück
`Maximum (p)`	gibt das größte Element des Teilbaums mit Wurzel `p` zurück
`Successor (p)`	gibt den Nachfolger von Knoten `p` in Inorder–Durchmusterungsreihenfolge zurück, falls dieser existiert, ansonsten `NIL`
`Predecessor (p)`	gibt den Vorgänger von Knoten `p` in Inorder–Durchmusterungsreihenfolge zurück, falls dieser existiert, ansonsten `NIL`

Aufgabe 4: Was macht der folgende Algorithmus? **15 Punkte**

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

```
(A0) Algorithmus MachIrgendwas (A, p, q)
(A1) /* Input: Folge von n Zahlen A[1, ..., n], p, q */
(A2) /* Output: ??? */
(A3)    Falls p < q {
(A4)       r = WhatAmI(A, p, q);
(A5)       MachIrgendwas(A, p, r − 1);
(A6)       MachIrgendwas(A, r + 1, q);
(A7)    }

(W0) Prozedur WhatAmI (A, p, q)
(W1)    x = A[q]
(W2)    i = p − 1
(W3)    Für j = p, p + 1, . . . , q {
(W4)       Falls A[j] ≤ x {
(W5)          i = i + 1;
(W6)          vertausche A[i] und A[j];
(W7)       }
(W8)    }
(W9)    Return i;
```

a.) Wie sieht der Output des Algorithmus `MachIrgendwas`$(A, 1, n)$ aus?

b.) Was macht die Prozedur `WhatAmI`(A, p, q)? Wie sieht das Feld $A[p, \ldots, q]$ nach Aufruf von `WhatAmI` aus? Was kann man über den Rückgabewert i sagen?

c.) Erklären Sie in eigenen Worten wie der Algorithmus `MachIrgendwas`$(A, 1, n)$ arbeitet.

d.) Geben Sie eine Eingabefolge A der Länge n für den Worst-Case bezüglich der Anzahl der Vertauschungen an (hierbei sind auch unechte Vertauschungen zu zählen, Sie können also eine Vertauschung als einen Aufruf der Zeile (W6) betrachten). Wieviele Vertauschungen entstehen dabei?

e.) Wie lautet die Laufzeit im Best–Case (Θ–Notation) und unter welchen Umständen tritt sie ein?

Aufgabe 5: Suchen in geordneten Feldern **10 Punkte**

Geben Sie eine Java–Implementierung für folgendes Problem an:

Gegeben ist eine ganze Zahl x und ein aufsteigend sortiertes Feld ganzer Zahlen `A[0]`, ..., `A[n-1]` der Länge n. Finde i, so dass `A[i]` $= x$ oder gib -1 zurück, wenn x nicht im Feld vorkommt.

Komplettieren Sie folgenden Code, so dass ein Aufruf der Funktion nur Zeit $O(\log n)$ benötigt.

```
public static int search (int[] A, int x)
{




}
```

Hinweis: Die Anzahl der Elemente eines Feldes `A` kann in Java mit `A.length` abgefragt werden.

Aufgabe 6: Hash-Verfahren **12 Punkte**

Gegeben sind die beiden folgenden Hashfunktionen:

$$h_1(k) = k \bmod 10 \quad (1)$$
$$h_2(k) = (k \bmod 2) + 2 \ . \quad (2)$$

a.) Wie lautet die Hash–Funktion $h(k, i)$ für die Methode Double–Hashing?

b.) Fügen Sie die folgenden Schlüssel mit Hilfe der Methode Double–Hashing in eine Hashtabelle der Größe 12 ein. Stellen Sie die Hashtabelle jeweils nach jedem Einfügeschritt dar.

34, 28, 45, 14, 80, 36, 58, 65

c.) Sind die beiden Hashfunktionen für die Double–Hashing–Methode geeignet? Falls nicht, schlagen Sie zwei gute Hashfunktionen für dieses Problem vor. Begründen Sie Ihre Antwort.