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 18. Dezember 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	14	14	15	12	15	10	80	_____
erreichte Punktzahl								

Viel Erfolg!

Aufgabe 1: O-Notation **14 Punkte**

a.) Seien $f(n)$ und $g(n)$ Funktionen mit positivem Wertebereich. Sind die folgenden Aussagen wahr oder falsch?

Diese Aussage ist	wahr	falsch
Aus $f(n) = \mathrm{O}(g(n))$ folgt $g(n) = \mathrm{O}(f(n))$.		
Aus $f(n) = \mathrm{O}(g(n))$ folgt $20f(n) = \mathrm{O}(\frac{1}{2}g(n))$.		
Aus $f(n) = \mathrm{O}(5g(n))$ folgt $g(n) = \Omega(5f(n))$.		
Aus $f(n) = \Theta(g(n))$ folgt $g(n) = \Omega(f(n))$.		
Aus $f(n) = \Omega(g(n))$ folgt $f(n) = \Omega(\frac{n}{1000}g(n))$.		
$f(n) + g(n) = \Theta(\min(f(n), g(n)))$.		
$f(n) = \mathrm{O}((f(n)^2)$.		
$f(n) = \Theta(f(\frac{n}{2}))$.		
Aus $f(n) = \mathrm{O}(g(n))$ folgt $\log_2(f(n)) = \mathrm{O}(\log_2(g(n)))$ für $\log_2(g(n)) > 0$ und $f(n) \geq 1$.		

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

$$f(n) = \left\{ \begin{array}{ll} 10 \log n & \text{für } n > 1000 \\ \frac{1}{1000}n^2 - 1000n & \text{für } n \leq 1000 \end{array} \right\}$$

Ist $f(n) = \mathrm{O}(n \log n)$? Beweisen oder widerlegen Sie diese Aussage.

Aufgabe 2: Sortierverfahren: Merge–Sort **14 Punkte**

a.) Geben Sie den Pseudocode von *Merge-Sort*(A, p, q) an. Halten Sie sich an folgende Bezeichnungen: Die N-elementige Folge ist in dem Array $A[1..N]$ gespeichert, und p und q geben den linken und rechten Index an. Nehmen Sie dabei an, dass die Prozedur Merge(A, p, q, m) bereits gegeben ist. (Es ist also nicht notwendig den Pseudocode hierfür anzugeben.)

b.) Führen Sie das Sortierverfahren *Merge-Sort* (passend zu Ihrem Pseudocode) mit der folgenden Folge durch:

21, 14, 28, 8, 17, 36, 15, 18, 5, 2

Geben Sie dabei für jeden Aufruf der Prozedur **Merge(**A, p, q, m**)** die Indexgrenzen p, q und m sowie die Folge A **vor** und **nach** dem Aufruf an.

Aufgabe 3: Durchmusterung von binären Bäumen **15 Punkte**

a.) Gegeben sind die Preorder und die Inorder-Reihenfolge eines binären Baumes T:

Preorder-Reihenfolge:	c, k, g, a, l, d, b
Inorder-Reihenfolge:	k, c, d, l, a, b, g

(a) Konstruieren Sie den dazugehörigen binären Baum T.

(b) Wie lautet die Postorder-Reihenfolge von T?

b.) Geben Sie den Pseudocode eines Algorithmus **LevelDurchmusterung** an, der einen binären Baum in Level-Reihenfolge durchläuft.

Die Level-Reihenfolge eines binären Baums T ist gegeben durch: Durchsuche zuerst alle Knoten der Stufe 1 (die Wurzel des Baumes), dann alle Knoten der Stufe 2 (Kinder der Wurzel) von links nach rechts, danach alle Knoten der Stufe 3 (Kinder der Kinder der Wurzel) von links nach rechts, usw.

Verwenden Sie hierzu die verallgemeinerte Listenstruktur, die über `p.key`, `p.left`, `p.right` ansprechbar ist.

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? **12 Punkte**

Gegeben sei ein Array $A[1, ..., n]$ mit n ganzen Zahlen und Indizes $i, j \in \{1, \ldots, n\}$.

(A00) **Algorithmus** `WhatAmI` (A, i, j)
(A01) /* **Input:** Folge von n Zahlen $A[1, ..., n]$ und $i, j \in \{1, \ldots, n\}$ */
(A02) /* **Output:** ??? */
(A03) Falls $i == j$ return(A[i]);
(A04) Falls $i == j - 1$
(A05) Falls $A[i] < A[j]$ return(A[j])
(A06) Sonst return(A[i]);
(A07) Sonst {
(A08) $p = (i + j)$ div 2;
(A09) $r_1 =$ `WhatAmI`(A, i, p);
(A10) $r_2 =$ `WhatAmI`$(A, p + 1, j)$;
(A11) Falls $r_1 < r_2$ return(r_2)
(A12) Sonst return(r_1);
(A13) }

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

b.) Was kann man über den Rückgabewert von `WhatAmI`(A, i, j) sagen?

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

d.) Welche Laufzeit hat der Algorithmus im schlechtesten Fall (in O-Notation) für eine n-elementige Folge? Dabei können Sie annehmen, dass $n = 2^k$ für ein $k > 0$. Erläutern Sie Ihre Antwort.

e.) Ist `WhatAmI` ein geeigneter Algorithmus? Falls ja, begründen Sie Ihre Antwort, falls nein, schlagen Sie eine Alternative vor.

Aufgabe 5: Binäre Suchbäume **15 Punkte**

Geben Sie eine Java–Implementierung für folgendes Problem an:

Gegeben sei ein binärer Suchbaum B mit ganzzahligen Schlüsseln. Gegeben sei außerdem ein Schlüssel x, der in dem Baum enthalten ist. Gesucht ist der kleinste Schlüssel in B, der echt größer als x ist.

Geben Sie eine Java-Implementierung 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 größerer im Binärbaum gespeicherter Schlüssel stets vorkommt, da im Binärbaum ein unechter Schlüssel mit sehr großem Wert 100000 gespeichert ist.

Verwenden Sie hierzu die folgende Klasse `binarySearchTree` und komplettieren Sie die Methode `search`. Achten Sie darauf, aussagekräftige Bezeichner zu verwenden und versehen Sie Ihren Code mit Kommentaren.

Klassendefinition:

```
class node {
    int key;                    // Schluessel
    node left, right, father;  // linkes Kind, rechtes Kind, Vater
}

class binarySearchTree {
    static final int inf = 100000;
    node root;

    ... // andere Methoden

    public int search(int x) {
      .
      .
      .
    }

}
```

Aufgabe 6: Komplexität: Heap und Hashing **10 Punkte**

a.) **Datenstruktur Heap**

Gegeben sind n ganzzahlige Schlüssel, die in der Datenstruktur Heap gespeichert sind. Die größte Zahl sei an der Wurzel gespeichert. Geben Sie die Komplexität in O-Notation der folgenden Operationen auf dem Heap im schlechtesten Fall an. Am Ende jeder Operation soll stets ein Heap zurückbleiben.

Operation	Komplexität
Einfügen eines beliebigen Elementes	
Suchen des Maximums	
Suchen eines beliebigen Elementes	
Entfernen eines beliebigen Elementes	
Suchen des Minimums	
Entfernen des Maximums	

b.) **Hashing–Verfahren**

(a) Wieviele Schritte (in O-Notation) werden im schlechtesten Fall benötigt, um in eine leere Hashtabelle n Schlüssel einzufügen, wenn zur Überlaufbehandlung die Methode der Verkettung der Überläufer als verkettete lineare Liste

i. mit **unsortierten** Listen verwendet wird?

ii. mit **sortierten** Listen verwendet wird?

(b) Wieviele Schritte benötigt man insgesamt, um nach jedem der n eingefügten Schlüssel einmal zu suchen, wenn zur Überlaufbehandlung die Methode der Verkettung der Überläufer als verkettete lineare Liste

i. mit **unsortierten** Listen verwendet wird?

ii. mit **sortierten** Listen verwendet wird?

Begründen Sie in jedem Fall Ihre Antworten.