PRÜFUNGSORDNER - ein Service Deiner Fachschaft Informatik!
Algodat I (Mutzel)
Preis: 11/0,80€

Institut für Computergraphik und Algorithmen

Abteilung für Algorithmen und Datenstrukturen

TU TECHNISCHE UNIVERSITÄT WIEN

Prof. Petra Mutzel
Gunnar Klau
Ivana Ljubic
René Weiskircher

Wintersemester 2000/2001

Klausur zur Vorlesung Algorithmen und Datenstrukturen 1

29. Jänner 2001

a.) Machen Sie bitte die folgenden Angaben in deutlicher Blockschrift:

Name: _____ Vorname: _____

Matrikelnummer: _____ Studienkennzahl: _____

b.) Legen Sie während der Klausur Ihren Studentenausweis vor sich auf das Pult.

c.) 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!

d.) Denken Sie daran, dass keinerlei Hilfsmittel erlaubt sind – weder Taschenrechner, irgendwelche Unterlagen, Handys,...

VOR DER ABGABE AUSZUFÜLLEN:

e.) Geben Sie bitte die Anzahl der zusätzlich abgegebenen Blätter an: _____

f.) 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		Note
bearbeitet							—	———
maximale Punktzahl	10	13	13	14	10	10	70	———
erreichte Punktzahl								

Viel Erfolg!

Aufgabe 1: Laufzeitabschätzungen (10 Punkte)

a.) Welche Laufzeit in Θ-Notation für n haben folgende Programme?

A
```
(0) Für i=1,...,n {
(1)   j = i + n*n;
(2)   printf("j");
(3) }
```

B
```
(0) Für i=1,...,n {
(1)   printf("i");
(2)   Für j=1,...,n {
(3)     sum = sum + a[i,j];
(4)   }
(5) }
```

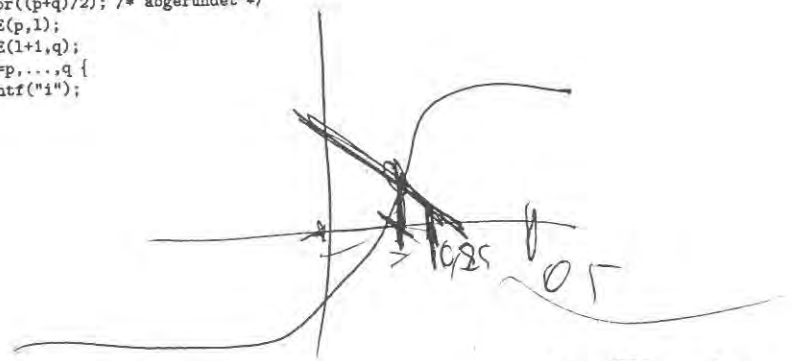
C
```
(0) sum = 0;
(1) Für i=1,...,n {
(2)   sum = sum + i;
(3) }
(4) Für i=1,...,sum {
(5)   printf("i");
(6) }
```

D
```
(0) n=2^k;   /* (2 hoch k) */
(1) Für i=1,...,k {
(2)   sum = sum + 2^k + n;
(3) }
```

Dabei zählen wir die Dauer der Berechnung von 2^k als 1.

E Aufruf von `NONAME(1,n)`:

```
(0) Algorithmus NONAME(p,q);
(1) Falls p<q {
(2)   l=floor((p+q)/2); /* abgerundet */
(3)   NONAME(p,l);
(4)   NONAME(l+1,q);
(5)   Für i=p,...,q {
(6)     printf("i");
(4)   }
(5) }
```



Aufgabe 2: Sortierverfahren **(13 Punkte)**

a.) Ein Sortieralgorithmus heißt stabil, wenn er so implementiert werden kann, dass die Reihenfolge von Elementen mit gleichem Schlüssel während des Sortierverfahrens nicht vertauscht wird. Welche der folgenden Sortierverfahren sind stabil?

Sortierverfahren	stabil	nicht stabil
Insertion Sort	✓	
Selection Sort		✗
Merge Sort		✓
Quick Sort		✗
Heap Sort		

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.) Sortieren Sie die Folge

$$314, 263, 329, 298, 193, 223, 253, 110$$

mit dem Sortierverfahren `Selection Sort`. Stellen Sie die Teilfolge nach jeder Iteration dar. Welche Datenstruktur ziehen Sie bei diesem Sortierverfahren vor: Listen oder Felder? Begründen Sie Ihre Antwort.

110
110, 193, 263, 253, 314

Aufgabe 3: Binäre Suchbäume **(13 Punkte)**

a.) Gegeben ist ein binärer Suchbaum mit Schlüsseln im Bereich von $[1, \ldots, 1000]$. Wir suchen nach dem Schlüssel mit der Nummer 363. Welche der folgenden Suchsequenzen können nicht durch die Suche in einem korrekten binären Suchbaum entstanden sein?

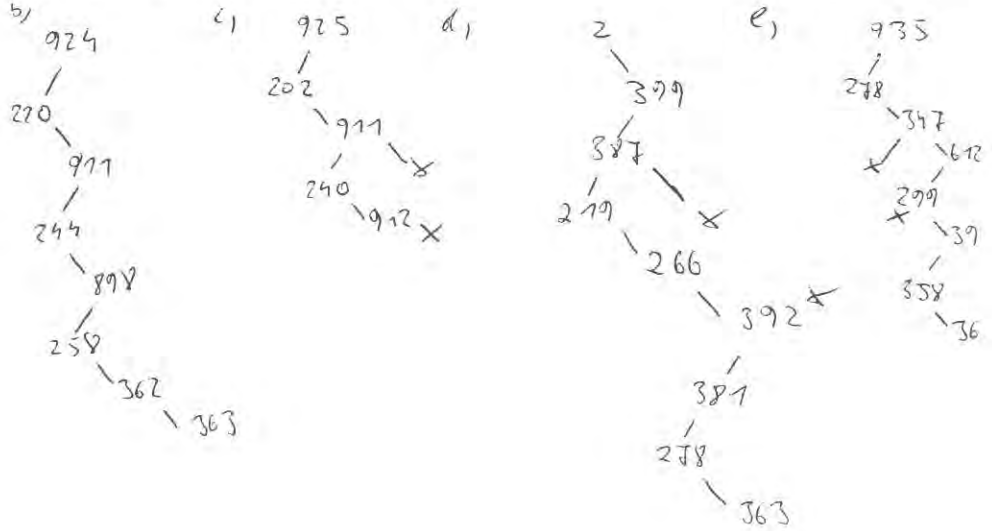
(a) $2, 252, 401, 398, 330, 344, 397, 363$ ✓
(b) $924, 220, 911, 244, 898, 258, 362, 363$ ✓
(c) $925, 202, 911, 240, 912, 245, 363$ ✗
(d) $2, 399, 387, 219, 266, 392, 381, 278, 363$ ✗
(e) $935, 278, 347, 621, 299, 392, 358, 363$ ✗

Begründen Sie Ihre Antwort.

b.) Gegeben ist ein binärer Suchbaum. Entwerfen Sie einen nichtrekursiven Algorithmus (Pseudocode), der einen binären Suchbaum mit Hilfe der unten angegebenen Prozeduren `Minimum` und `Successor` in `Inorder`-Reihenfolge durchläuft und jeweils den Schlüssel des besuchten Knotens ausgibt.

```
(0) Algorithmus Minimum(p)
(1)   Solange (p.leftson! = NULL)
(2)     p = p.leftson
(3)   Return p;

(0) Algorithmus Successor(p)
(1)   Falls (p.rightson! = NULL)
(2)     Return Minimum(p.rightson)
(3)   q = p.father;
(4)   Solange ((q! = NULL) und (p==q.rightson)) {
(5)     p=q; q=q.father; }
(6)   Return q;
```



Aufgabe 4: Was macht der folgende Algorithmus? **(14 Punkte)**

Gegeben sei ein Array $A[1,...,n]$ mit n ganzen Zahlen und der Eigenschaft $A[i] \leq A[i+1]$ für alle $i = 1,\ldots,n-1$. Weiterhin seien die ganzzahligen Variablen x und y global definiert und mit Anfangswert 0 belegt.

```
(0) Algorithmus WhatAmI (int A[1..n], int i)
(1) /* Input: Sortierte Folge von n Zahlen A[1,...,n] */
(2) /* Output: ??? */
(3)   Falls i == 1 {
(4)     x = 1; y = A[1]; }
(5)   Sonst {
(6)     WhatAmI (A,i-1);
(7)     Falls (A[i] == A[i - x]) {
(8)       x = x + 1; y = A[i]; }
(9)   }
```

a.) Was können Sie nach Ablauf des Algorithmus mit Aufruf WhatAmI (A,n) über die Werte der globalen Variablen x und y sagen?

b.) Was können Sie nach Ablauf des Algorithmus mit Aufruf WhatAmI (A,n) über die Werte der globalen Variablen x und y sagen, wenn das Feld A lauter gleiche Zahlen enthält?

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

d.) Geben Sie die Laufzeit des Algorithmus für den Worst-Case und den Best-Case in Θ-Notation an.

LIFO Last in First out

Aufgabe 5: Queue mit einfach verketteter Liste **(10 Punkte)**

Es geht darum, eine Queue mittels einer einfach verketteten Liste in JAVA zu implementieren. Die Konstruktoren sowie die Operationen `size` und `isEmpty` haben wir schon für Sie implementiert. Den Code für die Klasse `Node` und die schon implementierten Funktionen der Klasse `queue` finden Sie am Ende der Aufgabe.

a.) Geben Sie die Implementierung der beiden Funktionen `enqueue` und `dequeue` in JAVA an. Ihre Funktionen müssen zu den von uns zur Verfügung gestellten Code-Stücken passen. Die Funktion `enqueue` hängt ein neues Element an das Ende (Tail) der Queue an, während die Funktion `dequeue` das Element am Anfang der Queue (Head) löscht und zurückliefert. Ihre Funktionen müssen die in unserem Code angegebenen Rückgabewerte und Argumente haben.

b.) Begründen Sie, warum es besser ist, am Kopf (Head) der Liste Elemente zu entfernen und am Ende (Tail) der Liste einzufügen als am Ende zu löschen und am Anfang einzufügen.

```
class Node {
  private Object element;
  private Node next;
  Node() {
    this(null,null);
  }
  public Node(Object e, Node n) {
    element = e;
    next = n;
  }
  void setElement(Object newElem) {element = newElem;}
  void setNext(Node newNext) {next = newNext;}
  Object getElement() { return element;}
  Node getNext() {return next;}
}

public class queue {
  private Node head;
  private Node tail;
  private int size;
  public queue() {
    head = null;
    tail = null;
    size = 0;
  }
  public int size() {
    return size;
  }
  public boolean isEmpty() {
    return (head == null);
  }
  public void enqueue(Object obj) {
        . . .
  }
  public Object dequeue() {
        . . .
  }
}
```

Stack : Last in First out

Aufgabe 6: Hashing **(10 Punkte)**

a.) Fügen Sie die folgenden Zahlen in eine Hashtabelle der Größe 15 ein.

10,8,23,20,3,5,12,17,19,9

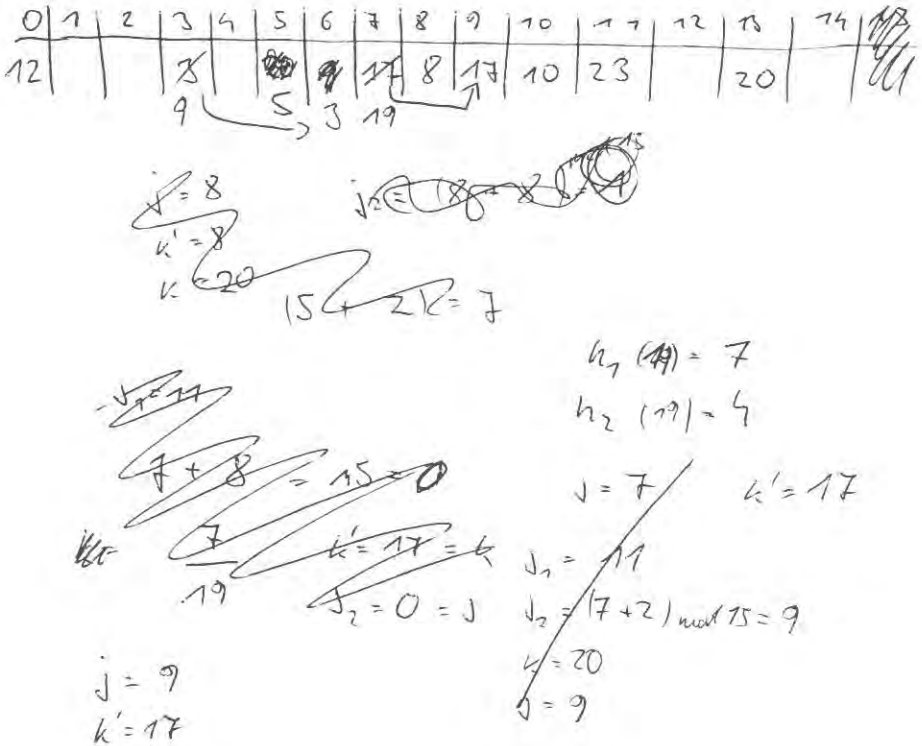
Verwenden Sie zur Kollisionsbehandlung Double Hashing mit dem Algorithmus von Brent mit den beiden Hashfunktionen $h_1(k)$ und $h_2(k)$:

$$h_1(k) = k \bmod 12 \quad (1)$$
$$h_2(k) = k \bmod 15 \quad (2)$$

Stellen Sie die Hashtabelle nach jedem Einfügen dar.

b.) Sind $h_1(k)$ und $h_2(k)$ geeignete Hashfunktionen? Begründen Sie Ihre Antwort.

c.) Geben Sie zwei alternative Hashfunktionen für Double Hashing mit Hilfe von Brent an, die für eine Hashtabelle der Größe 15 gut geeignet sind.



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	Σ	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 $f(n) = O(g(n))$ folgt $g(n) = O(f(n))$.		
Aus $f(n) = O(g(n))$ folgt $20f(n) = O(\frac{1}{2}g(n))$.		
Aus $f(n) = O(5g(n))$ 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 $f(n) = \Omega(\frac{n}{1000}g(n))$.		
$f(n) + g(n) = \Theta(\min(f(n), g(n)))$.		
$f(n) = O((f(n)^2)$.		
$f(n) = \Theta(f(\frac{n}{2}))$.		
Aus $f(n) = O(g(n))$ folgt $\log_2(f(n)) = O(\log_2(g(n)))$ 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 $f(n) = O(n \log n)$? 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 ∈ {1, ..., 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)     r1 = WhatAmI(A, i, p);
(A10)     r2 = WhatAmI(A, p + 1, j);
(A11)     Falls r1 < r2     return(r2)
(A12)     Sonst             return(r1);
(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.

Institut für Computergraphik
E186/1 Informatik

Abteilung für Algorithmen und Datenstrukturen

TU TECHNISCHE UNIVERSITÄT WIF

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

Sommerseme

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 Papi ausgeht, bitten Sie die Aufsicht um Nachschub. Es ist nicht erlaubt eigenes Papier zu verwen den!
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 :

Aufgabe	A 1	A 2	A 3	A 4	A 5	A 6	Σ	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.