Lehrstuhl Praktische Informatik III
Universität Mannheim
Prof. Dr. Guido Moerkotte
Thorsten Fiebig

März 2001

Vordiplomklausur
Algorithmen und Datenstrukturen
Wintersemester 2000/2001

Name: .....
Vorname: .....
Matrikelnummer: .....
Studienfach: .....

Hinweise:

1. Prüfen Sie Ihr Klausurexemplar auf Vollständigkeit.
2. Alle 10 Aufgaben sind zu bearbeiten.
3. Es sind keine Hilfsmittel zugelassen.
4. Die Klausur dauert 100 Minuten.
5. Beantworten Sie alle Fragen auf den Aufgabenblättern.
6. Bei einem Teil der Aufgaben erfolgt auf falsche Antworten ein Punkteabzug. Diese Aufgaben sind mit einem expliziten Hinweis in der Aufgabenstellung gekennzeichnet. Die Gesamtpunktzahl einer solchen Aufgabe kann nicht unter 0 Punkte sinken.

	maximale Anzahl Punkte	erreichte Anzahl Punkte
Aufgabe 1	4	
Aufgabe 2	12	
Aufgabe 3	28	
Aufgabe 4	6	
Aufgabe 5	8	
Aufgabe 6	6	
Aufgabe 7	12	
Aufgabe 8	8	
Aufgabe 9	6	
Aufgabe 10	9	
	99	

1. (4 Punkte)

   Sei $A[1, \ldots, n]$ ein Array der Länge n. In der folgenden Tabelle sind mehrere Sortieralgorithmen aufgeführt. Geben Sie für jeden Sortieralgorithmus die zum Sortieren von A benötigte Worst-Case-Laufzeit an. Verwenden Sie die Θ-Notation.

Insertion-Sort	
Heap-Sort	
Quick-Sort	
Bucket-Sort	

---

2. (12 Punkte)

   Die folgenden Rekurrenzen lassen sich alle mittels des Master-Theorems lösen. Kreuzen Sie für jede Rekurrenz den passenden Fall des Master-Theorems an.

Rekurrenz	1. Fall	2. Fall	3. Fall
$T(n) = 3T(n/7) + n^2$	☐	☐	☐
$T(n) = 2T(n/2) + \sqrt{n}$	☐	☐	☐
$T(n) = 3T(n/3) + n$	☐	☐	☐
$T(n) = 5T(n/3) + \lg n$	☐	☐	☐

3. (28 Punkte)

In der Vorlesung wurden Operationen auf einer Menge S vorgestellt, deren Elemente mit einem Schlüssel versehen sind. Auf der Schlüsselmenge ist eine totale Ordnung definiert.

Search(S,k):	sucht das Element mit dem Schlüssel k in der Menge S
Insert(S,x):	fügt das Element x in S ein
Delete(S,x):	löscht das Element x aus S
Minimum(S):	liefert das Element mit dem minimalen Schlüssel aus S
Maximum(S):	liefert das Element mit dem maximalen Schlüssel aus S
Successor(S,x):	liefert zu x das Element mit dem nächstgrößeren Schlüssel
Predecessor(S,x):	liefert zu x das Element mit dem nächstkleineren Schlüssel

(Beachten Sie, daß x einen Verweis auf ein Element darstellt und nicht den Schlüssel eines Elements!) Außerdem wurden verschiedene Möglichkeiten zur Mengenimplementierung vorgestellt. Die Laufzeit einer Mengenoperation ist abhängig von der Mengenimplementierung. In der folgenden Tabelle sollen nun in Abhängigkeit von den Mengenimplementierungen die Laufzeiten der Mengenoperationen eingetragen werden. Verwenden Sie die Θ-Notation. Gehen Sie davon aus, daß $|S| = n$ ist.

Tragen Sie in folgende Tabelle die Worst-Case-Laufzeiten ein.

	unsortierte einfach verkettete Liste	unsortierte doppelt verkettete Liste	Rot-Schwarzbaum	Binary Heap
Search(S,k)				
Insert(S,x)				
Delete(S,x)				
Minimum(S)				
Maximum(S)				
Successor(S,x)				
Predecessor(S,x)				

4. (6 Punkte)

Beantworten Sie folgende Fragen nur mit ja oder nein. **Auf eine falsche Antwort erfolgt Punkteabzug!**

(a) Erfüllt im allgemeinen Fall ein Heap die Binäre Suchbaumeigenschaft?

(b) Erfüllt im allgemeinen Fall ein Rot-Schwarzbaum die Heap-Eigenschaft?

(c) Gibt es einen Rot-Schwarzbaum, der die Heap-Eigenschaft erfüllt?

---

5. (8 Punkte)

Beantworten Sie folgende Fragen nur mit ja oder nein. **Auf eine falsche Antwort erfolgt Punkteabzug!**

(a) In der Vorlesung wurde das Aktivitätsauswahlproblem und dessen Lösung mittels eines Greedy-Algorithmus beschrieben. Das Mengensystem (A, S) zum Aktivitätsauswahlproblem besteht aus einer Menge von Aktivitäten A und einer unter Inklusion abgeschlossenen Teilmenge S der Potenzmenge von A. S enthält alle Teilmengen von A, die aus sich nicht überschneidenden Aktivitäten bestehen.

Bildet das Paar (A, S) ein Matroid?

(b) Auf einem ungerichteten Graph $G = (V, E)$ läßt sich ein Matroid $M = (E, I)$ über die Menge der Kanten definieren. Hierbei sei I eine unter Inklusion abgeschlossene Teilmenge der Potenzmenge von E, wobei jedes Element aus I nur die Kanten eines azyklischen Teilgraphen von G enthält.

Sei ω eine Gewichtsfunktion, die jede Kante aus E auf einen Wert in $\mathcal{R}^+$ abbildet. Sei weiterhin ω' eine Gewichtsfunktion, die jedes $i \in I$ auf einen Wert in $\mathcal{R}^+$ wie folgt abbildet:

$$\omega'(i) = \frac{\sum_{e \in i} \omega(e)}{|i|}.$$

Liefert GREEDY(M, ω') ein hinsichtlich ω' maximales Element aus I?

6. (6 Punkte)

Für die Hashtabelle $A[0, \ldots, 12]$ sei die Hashfunktion $h'(k) = k \bmod 13$ gegeben. Zur Kollisionsauflösung soll Offene Adressierung verwendet werden. Kreuzen Sie für die folgenden Hashfunktionen h an, ob für einen beliebigen Schlüssel k die Probensequenz $\langle h(k,0), h(k,1), \ldots, h(k,12)\rangle$ eine Permutation von $\langle 0, 1, \ldots, 12\rangle$ bildet. **Auf eine falsche Antwort erfolgt Punkteabzug!**

Hashfunktion	Erzeugt Permutation	Erzeugt keine Permutation
$h(k,i) = (h'(k) + i) \bmod 13$	☐	☐
$h(k,i) = (h'(k) + ih'(k)) \bmod 13$	☐	☐
$h(k,i) = (h'(k) + i(k \bmod 7)) \bmod 13$	☐	☐

---

7. (12 Punkte)

In der Vorlesung wurden die allgemeinen Lösungsstrategien *Greedy*, *Dynamic Programming* und *Divide and Conquer* vorgeschlagen. Wählen Sie zu jedem der folgenden Probleme eine Lösungsstrategie aus, mit der sich das Problem lösen läßt. Geben Sie keine Begründung an!

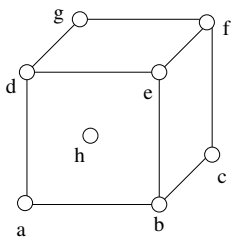
(a) Es soll in einem sortierten Integer-Array eine Zahl gesucht werden.

(b) Es sollen sehr große Zahlen multipliziert werden. Aufgrund ihrer Größe werden diese als Byte-Arrays mit variabler Länge dargestellt und können daher nicht mittels konstantem Aufwand multipliziert werden. Stattdessen soll der Aufwand einer Multiplikation zweier Zahlen, repräsentiert durch Arrays der Länge n und m, $\Theta(nm)$ betragen. Es gilt nun, für eine gegebene Menge von Zahlen einen vollständig geklammerten Multiplikationsterm mit minimalen Aufwand zu finden, der das Produkt der gegebenen Zahlen zum Ergebnis hat.

(c) Es seien eine Menge von ganzen Zahlen S und eine ganze Zahl M gegeben mit $\forall s \in S : s < M$. Es soll eine Teilmenge S' aus S bestimmt werden, mit $\sum_{s \in S'} = M$.

8. (8 Punkte)

Es sei der folgende ungerichtete Graph gegeben.



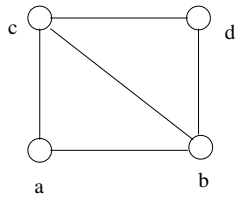
(a) Führen Sie eine Breitensuche ausgehend vom Knoten c durch. Gehen Sie dabei davon aus, daß die Adjazenzlisten alphabetisch sortiert sind. Geben Sie für jeden Knoten die Werte π und d an.

(b) Führen Sie eine Tiefensuche durch. Gehen Sie dabei davon aus, daß in der äußeren Schleife des DFS-Algorithmus die Knoten entsprechend der alphabetischen Reihenfolge besucht werden und daß die Adjazenzlisten alphabetisch sortiert sind. Geben Sie die Entdeckungszeit sowie die Endzeit für jeden Knoten an.

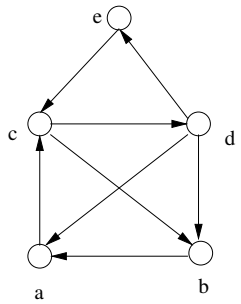
9. (6 Punkte)

In der Vorlesung wurden die Grapheigenschaften bipartit und azyklisch vorgestellt.

(a) Streichen sie aus folgendem ungerichteten Graphen eine minimale Anzahl von Kanten, um einen bipartiten Graphen zu erhalten.



(b) Streichen sie aus folgendem gerichteten Graphen eine minimale Anzahl von Kanten, um einen azyklischen Graphen zu erhalten.



---

10. (9 Punkte)

Konstruieren Sie den Musterautomaten für das Muster $P = ababa$ und das Alphabet $\Sigma = \{a, b, c\}$. Zeichnen Sie die Transitionen zwischen den vorgegebenen Zuständen ein.