UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Fakultät für Elektrotechnik, Informatik und Mathematik
Juniorprofessor Dr. Robert Elsässer

Klausur 2 - Datenstrukturen und Algorithmen SS 2009

Freitag, 02. Oktober 2009, 9-12Uhr

Name, Vorname: .....

Matrikelnummer: .....

Studiengang bitte ankreuzen:

- ☐ Informatik, Bachelor (seit WS 04/05)
- ☐ Informatik, Bachelor/Diplom (DPO 4)
- ☐ Informatik, Lehramt
- ☐ Ingenieurinformatik mit Schwerpunkt Informatik
- ☐ Ingenieurinformatik mit Schwerpunkt Elektrotechnik
- ☐ Ingenieurinformatik mit Schwerpunkt Maschinenbau
- ☐ Algorithmische Mathematik
- ☐ Mathematik Diplom
- ☐ Wirtschaftsinformatik
- ☐ Medienwissenschaften (80%)
- ☐ Magister mit Nebenfach Informatik (80%)
- ☐ Sonstiges: .....

Bitte Zutreffendes ankreuzen:

Fachprüfung: ☐ Leistungsnachweis: ☐ Teilnahmeschein: ☐

Bitte sorgfältig durchlesen und beachten!

- Prüfen Sie die Vollständigkeit Ihres Klausurexemplares (Deckblatt, 8 Aufgaben).
- Schreiben Sie zu Beginn der Klausur auf das Deckblatt und jede weitere Seite DEUTLICH LESBAR Ihren Namen und Ihre Matrikelnummer. KLAUSUREN OHNE DIESE ANGABEN WERDEN NICHT GEWERTET!
- Verwenden Sie keinen Rotstift, keinen Bleistift und kein eigenes Konzeptpapier. Benutzen Sie nur den Platz unter den Aufgabenstellungen und die Rückseiten der Blätter für Ihre Lösungen. Am Ende der Klausur befinden sich zwei Blätter Konzeptpapier.
- Sollten Sie zusätzliches Konzeptpapier benötigen, so wenden Sie sich bitte an die Aufsicht.
- Werden zu einer Aufgabe mehrere Lösungen angegeben, so gilt die Aufgabe als NICHT gelöst.
- Als Hilfsmittel ist NUR ein doppelseitig handbeschriebenes DIN A4 Blatt erlaubt; schreiben Sie auch auf dieses Blatt vor Beginn der Klausur Ihren Namen.
- **Zulassungsvorbehalt:** Hätten Sie wissen können, dass Sie zur Klausur nicht zugelassen sind, wird die bestandene Klausur NICHT gewertet; die nicht bestandene gilt ggf. als Fehlversuch.

Aufgabe	1	2	3	4	5	6	7	8	$\sum$
erreichbare Punkte	7	7	6	6	8	6	7	7	54
erreichte Punkte									

Name:	Matrikelnummer:

AUFGABE 1: Multiple-Choice (7 Punkte)
Hinweis: Es ist immer nur eine Antwort richtig. Falsche Antworten führen zu Punktabzügen!

1. In ungewichteten unzusammenhängenden Graphen eigenet sich BFS:
   ☐ zur Berechnung eines minimalen Spannbaums
   ☐ zum Sortieren
   ☐ zur Lösung des SSSP-Problems
   ☐ zur Berechnung eines Spannbaums

2. Der in der Vorlesung vorgestellte Dijkstra Algorithmus hat eine Laufzeit von:
   ☐ $\mathcal{O}((|V| + |E|)2^{\log\log|V|})$
   ☐ $\mathcal{O}(|V| \log |V|)$
   ☐ $\mathcal{O}(|E| \log |E|)$
   ☐ $\mathcal{O}((|V| + |E|) \log\log |V|)$

3. Der Algorithmus von Floyd-Warshall arbeitet nicht korrekt in:
   ☐ Graphen mit Zyklen
   ☐ nicht zusammenhängenden Graphen
   ☐ Graphen mit negativen Kantengewichten
   ☐ Graphen mit identischen Kantengewichten

4. Ein n-elementiger Heap besitzt auf einer bestimmten Höhe h höchstens
   ☐ $\log_h n$ Knoten
   ☐ $\log_{h+1} n$ Knoten
   ☐ $\left\lceil \frac{n}{2^h} \right\rceil$ Knoten
   ☐ $\left\lceil \frac{n}{2^{h+2}} \right\rceil$ Knoten

5. Welche Aussage ist nach dem Mastertheorem für die Rekursion $T(n) = 2 \cdot T(\frac{n}{2}) + n \log(n)$ korrekt?
   ☐ $\mathcal{O}(n)$
   ☐ $\mathcal{O}(\log(n))$
   ☐ $\mathcal{O}(n \log(n))$
   ☐ Das Mastertheorem kann nicht angewendet werden.

6. Die minimale Kodierungslänge des Textes "a b c d" mit $\Sigma = \{a, b, c, d\}$ ist
   ☐ 8 Bit
   ☐ 10 Bit
   ☐ 16 Bit
   ☐ 20 Bit

7. Welche der folgenden Aussagen ist richtig? Bei gierigen Algorithmen...
   ☐ ... wird die Lösung eines Problems aus mehreren zuvor berechneten Teillösungen zusammengesetzt.
   ☐ ... wird auf jede zulässige Lösung die Zielfunktion angewendet, um die optimale Lösung zu finden.
   ☐ ... wird eine bereits gefundene Teillösung schrittweise durch eine lokal optimale Wahl erweitert.
   ☐ ... werden Teillösungen eventuell mehrfach berechnet.

Name:	Matrikelnummer:

AUFGABE 2: Graphenalgorithmen (7 Punkte)
Sei $G = (V, E)$ ein ungerichteter (nicht zwangsweise zusammenhängender) Graph. Eine Kante $e \in E$ heisst *Brücke* in G, falls sich die Anzahl der im Graphen befindlichen Zusammenhangskomponenten durch das Entfernen von e vergrößert. Ein Beispiel für eine Brücke in einem Graphen ist in Abbildung 1 dargestellt.

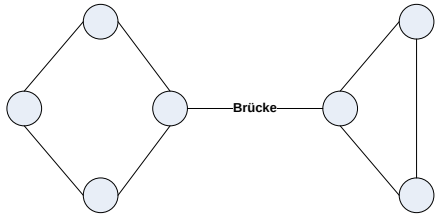

Abbildung 1: Beispielgraph mit einer Brückenkante

Entwickeln Sie einen Algorithmus A, der bei Eingabe eines ungerichteten Graphen $G = (V, E)$ in Zeit $\mathcal{O}(|V| \cdot |E| + |E|^2)$ entscheidet, ob der Graph G eine Brücke besitzt. Beschreiben Sie hierzu ihren Algorithmus kurz und präzise (aber vollständig!) in einer informellen Form; zeigen Sie die Korrektheit und Laufzeit. Sie müssen Ihren Algorithmus nicht in Pseudocode angeben, können dies jedoch zusätzlich tun.

Name:	Matrikelnummer:

Name:	Matrikelnummer:

AUFGABE 3: Randomisierung (6 Punkte)
Hinweis: $\forall k \in \mathbb{N}_{\geq 2} : \frac{1}{e} \geq \left(1 - \frac{1}{k}\right)^k \geq \frac{1}{4}$
Hinweis: Das Array A ist von 0 bis $n-1$ indiziert.
Bearbeiten Sie nachfolgende Teilaufgaben:

1. Gegeben sei ein Array A mit n paarweise verschiedenen Elementen und eine Operation $rand(0, n-1)$ (mit Ausführungszeit $\mathcal{O}(1)$), welche zufällig und gleichverteilt eine Zahl aus der Menge $\{0, ..., n-1\}$ ausgibt. Konstruieren Sie einen Algorithmus Alg, welcher in konstanter Zeit mit Wahrscheinlichkeit $\frac{1}{n}$ die Position von x im Array A findet. Sie können davon ausgehen, dass dieses Element in A an unbekannter Stelle vorhanden ist. Zeigen Sie, dass Alg das Element mit Wert x mit der gegebenen Wahrscheinlichkeit findet und analysieren Sie die Laufzeit.

2. Der Algorithmus aus a) wird nun so modifiziert, dass in einem Zeitschritt ein bestimmter Wert x mit genau $\frac{n}{100 \cdot \log n}$ zufällig und gleichverteilt gewählten Elementen aus A verglichen werden kann. Bei Auffinden des Elementes mit Wert x wird dessen Position erkannt und zurückgegeben. Wir nehmen hierbei an, dass $\frac{n}{100 \cdot \log n} \in \mathbb{N}$. Zeigen oder widerlegen Sie: es ist mit dem obigen Algorithmus nicht möglich das Element mit Wert x nach genau $\log n$ Zeitschritten mit konstanter Wahrscheinlichkeit zu finden.

3. Wenden Sie nachfolgenden Algorithmus auf die Eingabefolge $A = (16, 19, 49, 15, 11, 1, 7, 4, 3)$ an.

   Easy-Quicksort(A)
   x=rand(0,n-1) //bei der Lösung dieser Teilaufgabe wird hier das Ergebnis 4 geliefert
   A[x] $\leftrightarrow$ A[n-1]
   return *Partition(A,0,n-1)* //Algorithmus aus der Vorlesung

   Illustrieren Sie auf der nächsten Seite die einzelnen Schritte mitsamt der Arrayelemente und Zeigerpositionen des Partition-Algorithmus' anhand der vorgegebenen Schablonen.

Name:	Matrikelnummer:

Schritt	0	1	2	3	4	5	6	7	8	i	j
1)	16	19	49	15	11	1	7	4	3	-	-

Schritt	0	1	2	3	4	5	6	7	8	i	j
2)											

Schritt	0	1	2	3	4	5	6	7	8	i	j
3)											

Schritt	0	1	2	3	4	5	6	7	8	i	j
4)											

Schritt	0	1	2	3	4	5	6	7	8	i	j
5)											

Schritt	0	1	2	3	4	5	6	7	8	i	j
6)											

Schritt	0	1	2	3	4	5	6	7	8	i	j
7)											

Schritt	0	1	2	3	4	5	6	7	8	i	j
8)											

Schritt	0	1	2	3	4	5	6	7	8	i	j
9)											

Schritt	0	1	2	3	4	5	6	7	8	i	j
10)											

Schritt	0	1	2	3	4	5	6	7	8	i	j
11)											

Name:	Matrikelnummer:

Ersatzschablone:

Schritt	0	1	2	3	4	5	6	7	8	i	j
1)	16	19	49	15	11	1	7	4	3	-	-

Schritt	0	1	2	3	4	5	6	7	8	i	j
2)											

Schritt	0	1	2	3	4	5	6	7	8	i	j
3)											

Schritt	0	1	2	3	4	5	6	7	8	i	j
4)											

Schritt	0	1	2	3	4	5	6	7	8	i	j
5)											

Schritt	0	1	2	3	4	5	6	7	8	i	j
6)											

Schritt	0	1	2	3	4	5	6	7	8	i	j
7)											

Schritt	0	1	2	3	4	5	6	7	8	i	j
8)											

Schritt	0	1	2	3	4	5	6	7	8	i	j
9)											

Schritt	0	1	2	3	4	5	6	7	8	i	j
10)											

Schritt	0	1	2	3	4	5	6	7	8	i	j
11)											

Name:	Matrikelnummer:

AUFGABE 4: Datenverwaltung (6 Punkte)
Gesucht ist eine Datenstruktur zur Verwaltung einer dynamischen Menge von 2-Tupeln (x, y) mit $x, y \in \mathbb{N}$. Die Datenstruktur soll die folgenden vier Operationen jeweils in worst-case Laufzeit $O(\log n)$ und average-case Laufzeit $O(1)$ bereitstellen. Die Tupel sind dabei n zufällig und gleichverteilt gewählte Tupel aus der Menge $\{1, ..., m\} \times \{1, ..., m\}$. Es gilt $m >> n$ und wir nehmen an, dass m nicht bekannt ist. Hierbei bezeichnet n die Anzahl der Tupel, die sich aktuell in der dynamischen Menge befinden.

- **Einfügen(x,y)**: Fügt das Tupel (x, y) in die Datenstruktur ein, falls es sich noch nicht in der dynamischen Menge befindet.
- **Löschen(x,y)**: Lösche (x, y), falls sich (x, y) in der Datenstruktur befindet. Ansonsten bleibt die Menge unverändert.
- **Suchen(x,y)**: Es wird genau dann true zurückgegeben, falls sich das Tupel (x, y) in der dynamischen Menge befindet.
- **Zählen(x)**: Es wird die Anzahl der Tupel zurückgegeben, deren erste Stelle gleich x ist.

Beschreiben Sie den Aufbau Ihrer Datenstruktur und deren Operationen. Zeigen Sie zudem die korrekte Arbeitsweise Ihrer Datenstruktur sowie die Einhaltung der geforderten Laufzeiten. Hierbei ist kein formaler Beweis oder Pseudocode gefordert.

Name:	Matrikelnummer:

AUFGABE 5: Dynamische Programmierung (8 Punkte)
Gegeben sei eine Menge von n Messbechern $M_1, \ldots, M_n$. Der Messbecher M_i hat das Volumen $V_i \in \mathbb{N}$ Litern. Mit Hilfe der Messbecher sollen genau $x \in \mathbb{N}$ Liter Wasser in einen Tank geschüttet werden. Dabei kann ein Messbecher immer nur komplett gefüllt und in den Tank geschüttet werden. Aus dem Tank kann kein Wasser mehr entnommen werden. Ein Messbecher kann natürlich mehrfach verwendet werden. Einer der Messbecher hat ein Volumen von genau einem Liter.
Gesucht ist ein Algorithmus, der liefert, wie viele Schüttvorgänge mindestens benötigt werden. *Beispiel:* Für $n = 3$, $V_1 = 1$, $V_2 = 20$, $V_3 = 50$ und $x = 61$ werden mindestens 4 Schüttvorgänge benötigt ($x = V_2 + V_2 + V_2 + V_1$).

1. Geben Sie die Rekursionsgleichung für $s[i, j]$ an. $s[i, j]$ ist die minimale Anzahl der Schüttvorgänge, um j Liter in den Tank zu füllen, wenn nur die Messbecher $M_1, \ldots M_i$ verwendet werden dürfen. Geben Sie auch den Induktionsanfang an.

2. Geben Sie den *Pseudocode* eines Algorithmus an, der $s[n, x]$ in Laufzeit $O(n \cdot x)$ berechnet. Das Ergebnis soll zum Schluss in der Variablen m stehen.

Name:	Matrikelnummer:

Name:	Matrikelnummer:

AUFGABE 6: Huffman Kodierung (6 Punkte)
Gegeben Sei der folgende Text im Alphabet $\Sigma = \{a, b, c, d\}$

d d b d c a c d

1. Geben Sie die Frequenz jedes Buchstabens an

Buchstabe	a	b	c	d
Frequenz				

2. Zeichnen Sie den Baum der optimalen Präfixkodierung des Textes. Beschriften Sie die Blätter und Kanten so, wie es in der Vorlesung gemacht wurde.

3. Geben Sie eine passende binäre Kodierung jedes Buchstabens an.

Buchstabe	a	b	c	d
Kodierung				

4. Geben Sie die binäre Kodierung des Textes an.

   Kodierung:_____

Name:	Matrikelnummer:

AUFGABE 7: Min-Sort (7 Punkte)
Gegeben sei der folgende Sortieralgorithmus, der eine Folge $A = (a_1, \ldots, a_n)$ aufsteigend sortiert.
MinSort(A, n)

```
(1) for i ← 1 to n do
(2)     m ← i
(3)     for j ← i + 1 to n do
(4)         if A[m] > A[j]
(5)           then m ← j
(7)     A[i] ↔ A[m]
```

- Analysieren Sie die Laufzeit des Algorithmus in Θ-Notation für den Worst-Case und Best-Case.
- Beweisen Sie die Korrektheit des Algorithmus. Hinweis: Sie benötigen eine Schleifeninvariante für die innere Schleife und eine für die äußere Schleife.

Name:	Matrikelnummer:

AUFGABE 8: Datenstrukturen (7 Punkte)
Wir wollen Neuwagen auf einer einspurigen Fähre verwalten. Dazu speichern wir von jedem Auto das auf die Fähre geladen wird, die Fahrgestellnummer FNr. Zusätzlich möchten wir die Reihenfolge, in der die Autos aufgeladen wurden, festhalten. Da wir nur eine Laderampe haben, kann ein Auto A nur abgeladen werden, wenn alle nach A aufgeladenen Autos zuvor abgeladen wurden. Unsere Datenstruktur soll die folgenden Operationen in den unten angegebenen worst-case Laufzeiten unterstützen.

- *aufladen(FNr)*: Fügt die Fahrgestellnummer FNr eines Autos in die Datenstruktur ein.
  Laufzeit: $O(\log n)$
- *first()*: Gibt die Fahrgestellnummer FNr des Autos aus, welches als letztes aufgeladen, bzw. als nächstes abgeladen wird.
  Laufzeit: $O(1)$
- *abladen()*: Nimmt, der Reihenfolge entsprechend, die Fahrgestellnummer FNr des nächsten Autos aus der Datenstruktur und gibt diese aus.
  Laufzeit: $O(1)$
- *abladePlatz(FNr)*: Berechnet wie viele Autos abgeladen werden müssen, bevor das Auto mit der Fahrgestellnummer FNr abgeladen werden kann und gibt jene Anzahl aus.
  Laufzeit: $O(\log n)$

1. Beschreiben Sie eine zu dieser Aufgabe passende Datenstruktur. Sie soll die obigen Operationen in der jeweils geforderten Laufzeit unterstützen. Gehen Sie davon aus, dass die Fahrgestellnummer a priori eindeutig festgelegt wurde und dass wir mit der Datenstruktur beliebig große Fähren unterstützen wollen. D.h. Sie kennen die Grösse der Fähre nicht vorab.

2. Beschreiben Sie nun die Vorgehensweise der einzelnen Operationen und warum die geforderten Laufzeiten eingehalten werden.

Name:	Matrikelnummer:

Name:	Matrikelnummer: