Prof. Dr. G. Schnitger Frankfurt, den 06.04.2009

Klausur
Algorithmentheorie

WS 2008/2009

Name: _____ Vorname: _____

Matrikelnummer: _____ Geburtsdatum: _____

Studiengang: _____

⇓ BITTE GENAU LESEN ⇓

Die Klausur besteht aus **4** Aufgaben, in denen maximal **100 Punkte** erreicht werden können. Die Klausur ist mit Sicherheit bestanden, wenn zusammen mit der Bonifikation aus den Übungen mindestens **50 Punkte** erreicht werden.

Bitte schreiben Sie oben auf **jede** Seite in Blockschrift Namen und Matrikelnummer. Überprüfen Sie, ob die Klausur aus insgesamt **13** durchnummerierten Seiten besteht. Bitte benutzen Sie die Rückseiten. Weitere Blätter sind ggf. erhältlich.

Schreiben Sie **nicht** mit Bleistift. Ein DIN A4 Blatt mit Notizen ist das einzige zugelassene Hilfsmittel.

Eine umgangssprachliche, aber **strukturierte** Beschreibung von Algorithmen ist völlig ausreichend.

Werden zu einer Aufgabe 2 Lösungen angegeben, so gilt die Aufgabe als nicht gelöst. Entscheiden Sie sich also immer für **eine** Lösung.

Die Klausur dauert 180 Minuten. Der Termin für die Einsichtnahme ist Donnerstag, der 09. April von 10-11:30 Uhr im SR 307.

! VIEL ERFOLG !

1a	1b	1c	1d	1e	2a	2b	3a	3b	4a	4b	Σ
6	4	5	5	5	10	15	10	15	10	15	100

bestanden	nicht bestanden

Bonifikation	Σ

Name:	Matrikelnummer:

AUFGABE 1 — Sortieren

(a, 6 Punkte) Welche der Sortierverfahren Quicksort, Counting Sort oder Radix Sort sind stabil?

Quicksort: ○ ja ○ nein, Countingsort: ○ ja ○ nein, Radixsort: ○ ja ○ nein

(b, 4 Punkte) Ist die folgende Aussage wahr oder falsch: Jeder Sortieralgorithmus, der ausschließlich benachbarte Schlüssel vertauscht, hat eine Laufzeit von $\Omega(n^2)$.

○ Wahr ○ Falsch

(c, 5 Punkte) Beschreiben Sie das Sortierverfahren Radix Sort.

Name:	Matrikelnummer:

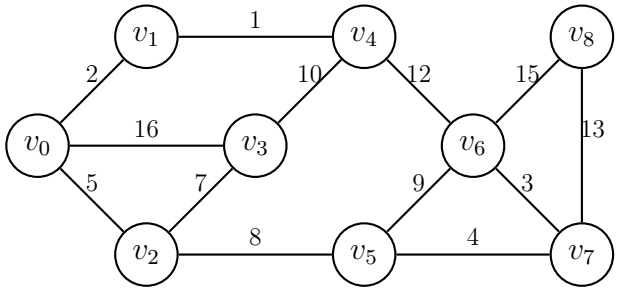
(d, 5 Punkte) n Zahlen $x_1, \ldots, x_n$ sind mit Radix Sort zu sortieren. Die natürliche Zahl k sei gegeben, und wir nehmen an, dass jedes x_i eine natürliche Zahl mit $0 \leq x_i \leq n^k$ ist. Welche Laufzeit besitzt Radix-Sort, wenn die Basis $b = n$ benutzt wird?

(e, 5 Punkte) Gegeben ist ein Array A mit n Schlüsseln, die alle zur Menge $\{1, 2, \ldots, n\}$ gehören. Beschreiben Sie einen möglichst schnellen Algorithmus, der für eine vorgegebene Zahl k die k kleinsten Schlüssel aus A ausgibt. Welche Laufzeit hat ihre Lösung?

Name:	Matrikelnummer:

AUFGABE 2 — Graphalgorithmen

(a, 5+5 Punkte) Gegeben sei der folgende Graph.



(1) Geben Sie an, welche Kanten der Algorithmus von Kruskal zur Bestimmung des minimalen Spannbaumes auswählt und in welcher Reihenfolge die Kanten ausgewählt werden.

(2) Geben Sie den vom Algorithmus von Dijkstra berechneten Baum der kürzesten Wege an. Startknoten ist der Knoten v_0.

Name:	Matrikelnummer:

(b, 15 Punkte) Gegeben ist ein ungerichter Graph $G = (V, E)$, dessen Kanten positive Gewichte besitzen. Modifizieren Sie Dijkstras Algorithmus, so dass für jeden Knoten $v \in V$ zusätzlich die *Anzahl* kürzester Wege vom Startknoten s nach v bestimmt wird. Eine Lösung in Zeit proportional zur Laufzeit von Dijkstra's Algorithmus ist möglich.

Name:	Matrikelnummer:

AUFGABE 3 — Entwurfsmethoden

(a, 10 Punkte) In der Vorlesung wurde das Rucksack Problem vorgestellt. Zur Erinnerung: Es sind n Objekte mit Gewichten $g_1, g_2, \ldots, g_n \in \mathbb{R}$ und Werten $w_1, w_2, \ldots, w_n \in \mathbb{N}_0$ sowie eine Gewichtsschranke G vorgegeben. Es ist eine Bepackung des Rucksacks zu finden, welche die Gewichtsschranke nicht überschreitet und den Wert der eingepackten Objekte maximiert.

Für eine Lösung des Rucksackproblems verwenden wir die Methode der dynamischen Programmierung. Wir setzen $\mathbf{W} = \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{w_i}$. Für jedes $1 \leq i \leq n$ und jeden möglichen Wert w mit $0 \leq w \leq W$ betrachten wir das Teilproblem

> Bestimme $Gewicht(i, w)$ = das minimale Gewicht einer Auswahl aus den ersten i Objekten, die einen Wert von genau w erreicht.
>
> Gibt es keine solche Bepackung mit Wert w, dann ist $Gewicht(i, w) = \infty$ zu setzen.

(1) Stellen Sie Rekursionsgleichungen auf, um ein Teilproblem $Gewicht(i, w)$ mit Hilfe der Teilprobleme $Gewicht(j, v)$ für $1 \leq j < i$ und $0 \leq v \leq w$ zu lösen.

(2) Wie kann die optimale Lösung des Rucksackproblems rekonstruiert werden, wenn bereits alle Teilprobleme gelöst wurden?

(3) In welcher Laufzeit wird die optimale Lösung des Rucksackproblems bestimmt?

Name:	Matrikelnummer:

(b, 15 Punkte) Im Problem der längsten gemeinsamen Teilfolge sind zwei Folgen $x = (x_1, x_2, \ldots, x_n)$ und $y = (y_1, y_2, \ldots, y_n)$ natürlicher Zahlen gegeben. Die Länge einer längsten gemeinsamen Teilfolge ist zu bestimmen. (Eine gemeinsame Teilfolge der Länge k wird durch Indizes $i_1, \ldots, i_k$, $j_1, \ldots, j_k$ mit $1 \leq i_1 < i_2 < \cdots < i_k \leq n$ und $1 \leq j_1 < j_2 < \cdots < j_k \leq n$ beschrieben. Die Indizes beschreiben eine gemeinsame Teilfolge genau dann, wenn $x_{i_1} = y_{j_1}, x_{i_2} = y_{j_2}, \ldots, x_{i_k} = y_{j_k}$ gilt.)

(1) Konstruieren Sie einen möglichst effizienten Algorithmus, der das Problem der längsten Teilsequenz löst. *Hinweis:* Wenden Sie das Verfahren der dynamischen Programmierung an und betrachten Sie „Präfixe“ $(x_1, x_2, \ldots, x_r)$ und $(y_1, y_2, \ldots, y_s)$.

(2) Begründen Sie die Korrektheit Ihres Algorithmus und analysieren Sie seine Laufzeit.

Name:	Matrikelnummer:

AUFGABE 4 — $\mathcal{NP}$-Vollständigkeit

(a, 10 Punkte) Beantworten Sie die folgenden Fragen mit Wahr oder Falsch. Eine Begründung ist **nicht** erforderlich. Es kann angenommen werden, dass $\mathcal{P} \neq \mathcal{NP}$ gilt.

Für jede richtige Antwort werden 2 Punkte vergeben. Eine falsche Antwort erhält -2 Punkte. Keine Antwort ergibt 0 Punkte. Die Mindestpunktzahl für diese Aufgabe ist 0 Punkte.

- Wenn $L_1 \leq_p L_2$ und wenn L_2 $\mathcal{NP}$-hart ist, dann ist auch L_1 $\mathcal{NP}$-hart.
  ○ Wahr ○ Falsch
- Wenn $L_1 \in \mathcal{NP}$ und $L_2 \subseteq L_1$, dann auch $L_2 \in \mathcal{NP}$.
  ○ Wahr ○ Falsch
- Jedes Problem L gehört entweder zur Klasse $\mathcal{P}$ oder zur Klasse $\mathcal{NP}$.
  ○ Wahr ○ Falsch
- Im Problem *Min-Spannbaum* ist ein ungerichteter Graph G gegeben, dessen Kanten Gewichte besitzen. Ebenso ist ein Schwellenwert S gegeben. Das Paar (G, S) gehört genau dann zu Min-Spannbaum, wenn G einen Spannbaum mit Gesamtgewicht höchstens S besitzt.

  Es existiert eine polynomielle Reduktion 3-SAT $\leq_p$ Min-Spannbaum von 3-SAT auf Min-Spannbaum.
  ○ Wahr ○ Falsch
- Das Problem

$$5\text{-}Clique = \left\{ G \;\middle|\; \begin{array}{l} G = (V, E) \text{ ist ein ungerichteter Graph} \\ \text{und } G \text{ enthält nur Cliquen der Größe höchstens } 5 \end{array} \right\}$$

  ist $\mathcal{NP}$-vollständig.
  ○ Wahr ○ Falsch

Name:	Matrikelnummer:

(b, 3+12 Punkte) Im Problem des *Multiprocessor Scheduling* sind gegeben:

- eine Menge von n Aufgaben, wobei die $i$te Aufgabe die Bearbeitungszeit $b(i)$ besitzt,
- m Maschinen sowie
- eine Frist F.

Es ist zu entscheiden, ob die n Aufgaben auf die m Maschinen verteilt werden können, sodass die Frist F eingehalten wird. (Die Frist F wird eingehalten, wenn jede Maschine nur Aufgaben mit Gesamtbearbeitungszeit höchstens F erhält.)

(1) Zeigen Sie, dass das Multiprocessor Scheduling Problem in $\mathcal{NP}$ liegt.

(2) Weisen Sie nach, dass *Multiprocessor Scheduling* $\mathcal{NP}$-vollständig ist.

Hinweis: Sie können annehmen, dass das Partition Problem $\mathcal{NP}$-vollständig ist. Im Partition Problem sind k natürliche Zahlen $a_1, \ldots, a_k$ gegeben, und es ist zu entscheiden, ob die k Zahlen in zwei Klassen mit gleicher Gesamtsumme aufgeteilt werden können, also:

$$PARTITION = \left\{ (a_1, a_2, \ldots, a_k) \;\middle|\; \begin{array}{l} \text{Es gibt eine Indexmenge } I \subseteq \{1, \ldots k\} \\ \text{mit } \sum_{i \in I} a_i = \sum_{j \in \{1,\ldots,k\} \setminus I} a_j \end{array} \right\}$$

Name:	Matrikelnummer:

Name:	Matrikelnummer:

Name:	Matrikelnummer:

Name:	Matrikelnummer: