

2. Klausur zur Vorlesung
Datenstrukturen und Algorithmen
SS 2008

Name :
Matrikelnummer :
Studiengang/Abschluss :

Die Klausur besteht aus acht Aufgaben. Die erreichbare Punktzahl zu jeder Aufgabe ist immer angegeben. Die maximal erreichbare Gesamtpunktzahl ist 54. Sie bestehen die Klausur, wenn Sie mindestens 27 Punkte erzielen.

Benutzen Sie **keinen Bleistift!** Bitte schreiben Sie oben auf jede Seite Ihren Namen und Ihre Matrikelnummer (bitte lesbar!).

Als Hilfsmittel ist lediglich ein beidseitig handschriftlich beschriebenes DIN-A4-Blatt zugelassen. Räumen Sie alles weitere (außer Ihrem Schreibmaterial) vom Tisch! Wer abschreibt **ODER** wer von sich abschreiben läßt, erhält die **Endnote mangelhaft (5,0)**. Bringen Sie Ihre Nachbarn also nicht in Schwierigkeiten!

Schreiben Sie Lösungen bitte unter die Aufgabenstellung. Reicht der Platz nicht aus, so benutzen Sie die Rückseite und die beigefügten Zusatzblätter. Weitere Blätter sind ggf. erhältlich.

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

Ergebnisse, Algorithmen und Datenstrukturen aus der Vorlesung dürfen zitiert werden. Alle weiteren Ergebnisse dürfen **nicht** zitiert werden und müssen neu hergeleitet werden. D.h. es reicht **nicht** zu sagen: „Das geht gemäß Aufg. 42 aus den Übungen.“

Sie haben bis 12:00 Uhr Zeit.

Viel Erfolg!

| | | | | | | | | | |
|---------|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | Σ |
| Punkte: | | | | | | | | | |

Klausur: Bonus: Note:

| | |
|-------|-----------------|
| Name: | Matrikelnummer: |
|-------|-----------------|

AUFGABE 1: (6 Punkte)

Kreuzen Sie die richtigen Antworten zu den folgenden Fragen an. Sie müssen Ihre Antwort dabei nicht begründen. Es gibt zu jeder Frage immer genau eine richtige Antwort.

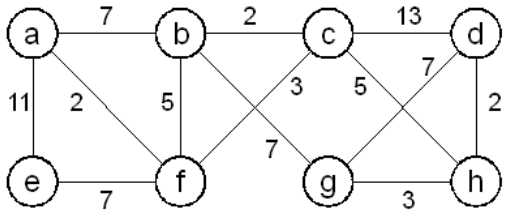
Jede richtige Antwort wird mit 1 Punkt, jede falsche Antwort mit -1 Punkt bewertet. Kreuzen Sie zu einer Frage mehr als eine Antwort ein, so gilt die Frage als falsch beantwortet. Wird keine Antwort angekreuzt, so wird die Frage mit 0 Punkten bewertet. Insgesamt können sie in dieser Aufgabe jedoch nicht weniger als 0 Punkte erhalten.

- a) Wie groß ist die average-case und die worst-case Laufzeit von INSERTION-SORT?
- average-case und worst-case $\Theta(n \log n)$
 - average-case $\Theta(n \log n)$ und worst-case $\Theta(n^2)$
 - average-case $\Theta(n^2)$ und worst-case $\Theta(n \log n)$
 - average-case und worst-case $\Theta(n^2)$
- b) Welche Rekursionsgleichung ergibt sich bei der Bestimmung der Laufzeit $T(n)$ des ALGORITHMUS VON STRASSEN zur Multiplikation zweier $n \times n$ Matrizen?
- $T(n) = 4T(n/2) + c_1n$
 - $T(n) = 7T(n/2) + c_2n^2$
 - $T(n) = 8T(n/2) + c_3n^2$
 - $T(n) = 8T(n/2) + c_4n^{2,81}$
- c) Welche Datenstruktur nutzt der *Algorithmus von Kruskal* zur Bestimmung eines minimalen Spannbaumes?
- Stack
 - Hashtabelle
 - Union-Find-Datenstruktur
- d) Welcher der folgenden Optimierungsalgorithmen ist *kein* gieriger Algorithmus (Greedy-Algorithmus)?
- Algorithmus von Floyd-Warshall (kürzeste Wege)
 - Algorithmus von Huffman (optimale Präfix-Kodierung)
 - Algorithmus von Prim (minimaler Spannbaum)
- e) Welche der folgenden asymptotischen Aussagen ist *falsch*?
- Für alle $a, b \geq 1$ gilt $\log_a(n) = \Theta(\log_b n)$.
 - Es existiert ein $\epsilon > 0$ mit $n^\epsilon = \Omega(\log(n))$.
 - Es existiert ein $k > 1$ mit $n = \mathcal{O}(\log^k(n))$.
- f) Welcher der drei Fälle aus dem *Master-Theorem* trifft auf die Rekursionsgleichung $T(n) = 2T(n/2) + n \log(n)$ zu?
- Es existiert ein $\epsilon > 0$ mit $f(n) = \mathcal{O}(n^{\log_b(a)-\epsilon})$.
 - Es gilt $f(n) = \Theta(n^{\log_b(a)})$.
 - Es existiert ein $\epsilon > 0$ mit $f(n) = \Omega(n^{\log_b(a)+\epsilon})$.
 - Keiner, $T(n)$ kann nicht mit dem Master-Theorem analysiert werden.

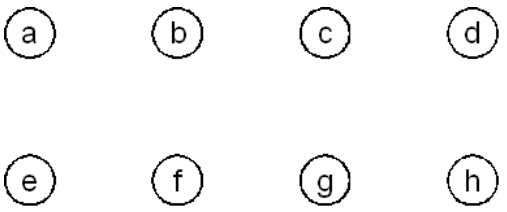
| | |
|-------|-----------------|
| Name: | Matrikelnummer: |
|-------|-----------------|

AUFGABE 2: (6 Punkte)

Gegeben sei der folgende gewichtete, ungerichtete Graph $G = (V, E)$. Mit Hilfe des *Algorithmus von Prim* mit Startknoten a wird ein minimaler Spannbaum von G bestimmt.



a) Geben Sie den resultierenden minimalen Spannbaum an.



b) Geben Sie die Reihenfolge an, in der die Kanten des minimalen Spannbaums gemäß Algorithmus bestimmt werden.

1. _____ 2. _____ 3. _____ 4. _____ 5. _____ 6. _____ 7. _____

c) Geben Sie nach jeder Auswahl einer Kante für den minimalen Spannbaum zu jedem Knoten $v \in V$ den Wert des in der verwendeten Heap-Datenstruktur gespeicherten Schlüssels $key[v]$ an. Lassen sie das Feld leer, wenn sich v nicht mehr im Heap befindet.

| | key[a] | key[b] | key[c] | key[d] | key[e] | key[f] | key[g] | key[h] |
|------------------|--------|--------|--------|--------|--------|--------|--------|--------|
| Initialisierung: | | | | | | | | |
| nach 1. Kante: | | | | | | | | |
| nach 2. Kante: | | | | | | | | |
| nach 3. Kante: | | | | | | | | |
| nach 4. Kante: | | | | | | | | |
| nach 5. Kante: | | | | | | | | |
| nach 6. Kante: | | | | | | | | |
| nach 7. Kante: | | | | | | | | |

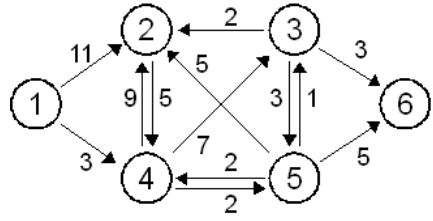
Name:

Matrikelnummer:

| | |
|-------|-----------------|
| Name: | Matrikelnummer: |
|-------|-----------------|

AUFGABE 3: (6 Punkte)

Gegeben sei der folgende gewichtete, gerichtete Graph (G, w) in Adjazenzlistendarstellung:



Es sollen die Gewichte der kürzesten Wege von Startknoten 1 zu jedem weiteren Knoten des Graphen $G = (V, E)$ mit dem ALGORITHMUS VON DIJKSTRA bestimmt werden. Dazu wird der Algorithmus mit dem Aufruf $\text{DIJKSTRA}(G, w, 1)$ gestartet.

```

DIJKSTRA( $G, w, s$ ):
1. for each  $v \in V \setminus \{s\}$  do  $d[v] \leftarrow \infty$ 
2.  $d[s] \leftarrow 0$ ;  $S \leftarrow \emptyset$ ;  $Q \leftarrow V$ 
3. while  $Q \neq \emptyset$  do
4.    $u \leftarrow \text{EXTRACT-MIN}(Q)$ 
5.    $S \leftarrow S \cup \{u\}$ 
6.   for each  $v \in \text{Adj}[u]$  do
7.     if  $d[v] > d[u] + w(u, v)$  then  $d[v] \leftarrow d[u] + w(u, v)$ 

```

Geben Sie den Inhalt des Arrays d nach der Initialisierung in den Schritten 1–2 sowie jeweils am Ende eines Durchlaufs der Schleife 3–7 an. Wir nehmen dabei an, dass die Adjazenzlisten von G aufsteigend sortiert sind und die Schleife 6–7 in eben dieser Reihenfolge durchlaufen wird.

| | d[1] | d[2] | d[3] | d[4] | d[5] | d[6] |
|----------------------------------|------|------|------|------|------|------|
| Initialisierung, nach Schritt 2: | | | | | | |
| Durchlauf 1, nach Schritt 7: | | | | | | |
| Durchlauf 2, nach Schritt 7: | | | | | | |
| Durchlauf 3, nach Schritt 7: | | | | | | |
| Durchlauf 4, nach Schritt 7: | | | | | | |
| Durchlauf 5, nach Schritt 7: | | | | | | |
| Durchlauf 6, nach Schritt 7: | | | | | | |

Name:

Matrikelnummer:

| | |
|-------|-----------------|
| Name: | Matrikelnummer: |
|-------|-----------------|

AUFGABE 4: (6 Punkte)

Die Laufzeit $T(n)$ eines Algorithmus sei gegeben durch die folgende Rekursionsgleichung:

$$T(n) = \begin{cases} 5 & \text{für } n = 1 \\ 2T(n/3) + 4n & \text{für } n > 1 \end{cases}$$

Analysieren Sie die Laufzeit $T(n)$ möglichst genau im \mathcal{O} -Kalkül mit Hilfe der Iterations- oder Substitutionsmethode. Sie dürfen dabei annehmen, dass n eine Potenz von 3 ist. Verwenden Sie *nicht* das Master-Theorem!

Hinweis: Bedenken Sie, dass Sie bei der Substitutionsmethode Ihr Ergebnis mit vollständiger Induktion beweisen müssen, während es bei der Iterationsmethode genügt, die allgemeine Form durch wiederholtes Einsetzen herzuleiten!

Name:

Matrikelnummer:

| | |
|-------|-----------------|
| Name: | Matrikelnummer: |
|-------|-----------------|

AUFGABE 5: (6 Punkte)

Gegeben Sei ein beliebiges Array A der Länge $n \geq 1$, welches die Max-Heap-Eigenschaft erfüllt. Wir nehmen dabei an, dass alle Elemente von A paarweise verschieden sind.

Beweisen Sie, dass es *keinen* Algorithmus geben kann, der nur mit Hilfe von Vergleichen und Zuweisungen in worst-case Laufzeit $\mathcal{O}(1)$ aus A das maximale Element entfernt und zugleich auf den verbleibenden $n - 1$ Elementen die Max-Heap-Eigenschaft wieder herstellt.

| | |
|-------|-----------------|
| Name: | Matrikelnummer: |
|-------|-----------------|

AUFGABE 6: (8 Punkte)

Gesucht ist eine Datenstruktur zur Verwaltung einer dynamischen Menge ganzer Zahlen beliebiger Größe. Die Datenstruktur soll die folgenden Operationen jeweils in worst-case Laufzeit $\mathcal{O}(\log n)$ unterstützen. Dabei bezeichnet n die Anzahl der Zahlen, die sich aktuell in der dynamischen Menge befinden.

- **EINFÜGEN(x):** Falls sich die Zahl x noch nicht in der dynamische Menge befindet, so wird x eingefügt. Ansonsten bleibt die dynamische Menge unverändert.
- **LÖSCHEMINIMUM():** Sofern die dynamische Menge nicht leer ist wird das Element mit dem kleinsten Wert aus der dynamischen Menge entfernt.
- **LÖSCHEMAXIMUM():** Sofern die dynamische Menge nicht leer ist wird das Element mit dem größten Wert aus der dynamischen Menge entfernt.
- **MITTELWERT():** Es wird der arithmetische Mittelwert (der Durchschnitt) aller Elemente in der dynamische Menge ausgegeben. Falls die dynamische Menge leer ist, so wird eine Fehlermeldung ausgegeben.

Beschreiben Sie in wenigen kurzen Sätzen, wie Ihre Datenstruktur aufgebaut ist und wie die angegebenen Operationen realisiert werden. Hierbei ist *kein Pseudocode* gefordert. Es sollte jedoch klar hervorgehen, dass ihre Datenstruktur korrekt arbeitet und die geforderten Laufzeitschranken eingehalten werden.

Name:

Matrikelnummer:

| | |
|-------|-----------------|
| Name: | Matrikelnummer: |
|-------|-----------------|

AUFGABE 7: (8 Punkte)

Einen gerichteten Graph $G = (V, E)$ nennen wir *schwach zusammenhängend*, wenn in G für je zwei Knoten $u, v \in V$ ein gerichteter Pfad von u nach v oder ein gerichteter Pfad von v nach u existiert.

- a) Beschreiben Sie einen Algorithmus, der bei Eingabe eines Graphen $G = (V, E)$ in Adjazenzlistendarstellung entscheidet, ob der Graph schwach zusammenhängend ist oder nicht. Ihr Algorithmus muss Laufzeit $\mathcal{O}(|V||E| + |V|^2)$ besitzen. Hierbei ist *kein Pseudocode* gefordert.
- b) Analysieren Sie die Laufzeit Ihres Algorithmus.
- c) Beweisen Sie die Korrektheit Ihres Algorithmus.

Name:

Matrikelnummer:

| | |
|-------|-----------------|
| Name: | Matrikelnummer: |
|-------|-----------------|

AUFGABE 8: (8 Punkte)

Eine Reederei möchte den Einsatz ihrer Frachtschiffsflotte für den Güterverkehr zwischen Hamburg und New York optimieren. Ihre Kunden erwarten, dass m Frachtcontainer (genormter Größe) den Zielhafen erreichen. Dazu stehen der Reederei eine Flotte F_1, F_2, \dots, F_n von n Frachtschiffen unterschiedlicher Bauart zur Verfügung, die nach Bedarf eingesetzt werden können. Das j -te Frachtschiff verfügt dabei über eine Ladekapazität von k_j Frachtcontainern und verursacht, sofern es eingesetzt wird, pauschal Betriebskosten in Höhe von b_j Euro. Aus Termingründen kann jedes Schiff der Flotte maximal einmal eingesetzt werden.

Gesucht ist ein Algorithmus, der nach der Methode der *dynamischen Programmierung* die minimalen Gesamtbetriebskosten ermittelt die notwendig sind, um alle m Container mit der zur Verfügung stehenden Flotte zum Zielhafen zu verschiffen. Sie können davon ausgehen, dass die Gesamtkapazität der Flotte ausreichend ist, um alle m Container zu transportieren.

- a) Bezeichne $S[i, j]$ die minimalen Gesamtbetriebskosten die benötigt werden, um i Container nur mit den Schiffen F_1, F_2, \dots, F_j zum Zielhafen zu transportieren. Geben Sie eine rekursive Formulierung für $S[i, j]$ an und beweisen Sie die Korrektheit Ihrer rekursiven Definition. Achten Sie darauf, auch alle Fälle der Rekursionsbasis vollständig und eindeutig zu definieren!
- b) Seien m und zwei Arrays $K = [k_1, k_2, \dots, k_n]$ und $B = [b_1, b_2, \dots, b_n]$ der Länge n gegeben, wobei k_j die Kapazität und b_j die Betriebskosten des j -ten Frachtschiffes angeben. Schreiben Sie einen Algorithmus *in Pseudocode*, der bei Eingabe (m, K, B, n) nach der Methode der *dynamischen Programmierung* die minimalen Gesamtkosten bestimmt. Die konkrete Auswahl der Schiffe muss dabei *nicht* explizit ausgegeben werden.
- c) Analysieren Sie die Laufzeit Ihres Algorithmus.

Name:

Matrikelnummer: