

1. 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 beigegefügteten 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.

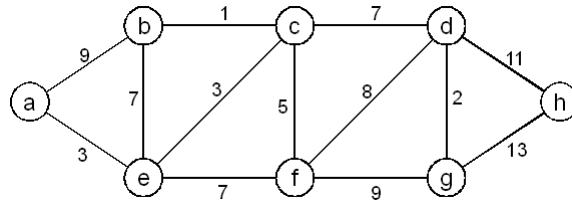
- a) Wie groß ist die average-case und die worst-case Laufzeit von MERGE-SORT?
- average-case und worst-case $\Theta(n \log n)$
 - average-case $\Theta(n \log n)$ und worst-case $\Theta(n^2)$
 - average-case $\Theta(n^2)$ und worst-case $\Theta(n \log n)$
 - average und worst-case $\Theta(n^2)$
- b) Welche Laufzeit benötigt COUNTING-SORT, um n Zahlen aus der Menge $\{1, 2, \dots, k\}$ zu sortieren?
- $\Theta(n \log n)$
 - $\Theta(n + k)$
 - $\Theta(n \cdot k)$
 - $\Theta(n^2)$
- c) Welche Datenstruktur nutzt der *Algorithmus von Dijkstra* zur effizienten Lösung des kürzeste-Wege-Problems (single source shortest path)?
- Stack („FIFO“)
 - Queue („LIFO“)
 - Heap
 - Hashtabelle
- d) Wie berechnet eine Implementierung nach der Methode der *dynamischen Programmierung* die optimale Lösung eines Problems mit rekursiver Struktur?
- rekursiv, „top-down“
 - iterativ, „bottom-up“
- e) Welche der folgenden asymptotischen Aussagen ist *falsch*?
- Für alle $k, \epsilon > 0$ gilt $\log^k(n) = \mathcal{O}(2^{\epsilon n})$.
 - Es existiert ein $\epsilon > 0$ mit $\log(n) = \Omega(n^\epsilon)$.
 - Es existiert ein $k \geq 1$ mit $5^{\log(n)} = \Theta(n^k)$.
- f) Welcher der drei Fälle aus dem *Master-Theorem* trifft auf die Rekursionsgleichung $T(n) = 2T(n/4) + \sqrt{3n}$ 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:

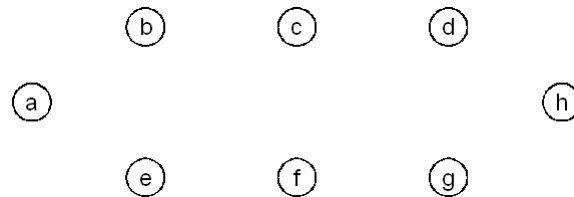
AUFGABE 3: (6 Punkte)

Gegeben sei der folgende gewichtete, ungerichtete Graph $G = (V, E)$.



Mit Hilfe des *Algorithmus von Kruskal* 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 schrittweise die disjunkten Mengen der verwendeten Union-Find-Datenstruktur nach jeder Auswahl einer Kante für den minimalen Spannbaum an. Markieren Sie dabei insbesondere die Repräsentanten der disjunkten Mengen.

Name:

Matrikelnummer:

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

AUFGABE 4: (6 Punkte)

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

$$T(n) = \begin{cases} 2 & \text{für } n = 1 \\ 4T(n/2) + 10n^2 & \text{für } n > 1 \end{cases}$$

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 Zweierpotenz 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)

Beweisen Sie, dass es *keinen* Algorithmus geben kann, der die folgenden drei Eigenschaften gleichzeitig erfüllt:

- Der Algorithmus benutzt wie ein Vergleichssortierer nur Vergleiche, Zuweisungen, usw.
- Der Algorithmus erstellt aus einem Array A einen binären Suchbaum mit den Elementen aus A .
- Bei Eingabe eines Arrays A der Länge n hat der Algorithmus Laufzeit $\mathcal{O}(n)$.

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

AUFGABE 6: (8 Punkte)

Gesucht ist eine Datenstruktur zur Verwaltung einer dynamischen Menge natürlicher Zahlen beliebiger Größe. Die Datenstruktur soll die drei folgenden Operationen jeweils in *average-case* Laufzeit $\Theta(1)$, sowie 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.
- LÖSCHEN(x): Falls sich eine Zahl x in der dynamische Menge befindet, so wird sie aus der dynamischen Menge entfernt. Ansonsten bleibt die dynamische Menge unverändert.
- SUCHEN(x) Es wird genau dann *true* ausgegeben, wenn sich die Zahl x in der dynamischen Menge befindet. Ansonsten wird *false* zurückgeliefert.

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 aus Ihrer Beschreibung hervorgehen, dass Ihre Datenstruktur korrekt arbeitet und die geforderten Laufzeitschranken eingehalten werden.

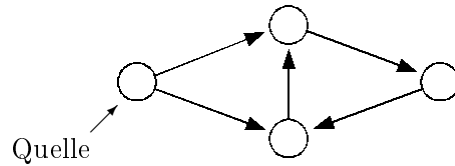
Name:

Matrikelnummer:

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

AUFGABE 7: (8 Punkte)

Ein Knoten v eines gerichteten Graphen heißt *Quelle*, wenn v Eingangsgrad 0 besitzt und für jeden weiteren Knoten w des Graphen ein gerichteter Pfad von v nach w existiert. Ein Beispiel eines Graphen mit Quelle ist in der folgenden Abbildung gegeben.



- Beschreiben Sie einen Algorithmus, der bei Eingabe eines Graphen $G = (V, E)$ in Adjazenzlistendarstellung in Laufzeit $\mathcal{O}(|V|^2 + |E||V|)$ überprüft, ob G eine Quelle besitzt oder nicht. Hierbei ist *kein Pseudocode* gefordert.
- Analysieren Sie die Laufzeit Ihres Algorithmus.
- Beweisen Sie die Korrektheit Ihres Algorithmus.

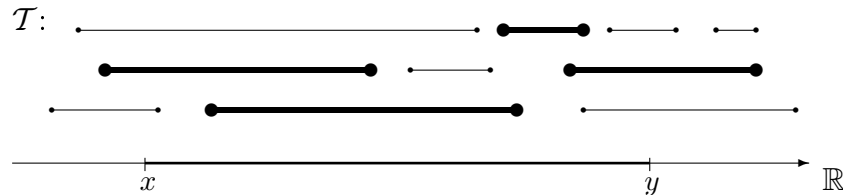
Name:

Matrikelnummer:

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

AUFGABE 8: (8 Punkte)

Wir betrachten das Problem *Intervallüberdeckung*: Gegeben sei ein Intervall $[x, y]$ in den reellen Zahlen sowie eine Menge von Teilintervallen $\mathcal{T} = \{[s_1, f_1], [s_2, f_2], \dots, [s_n, f_n]\}$. Unter einer *Überdeckung* des Intervalls $[x, y]$ verstehen wir eine Teilmenge der Teilintervalle aus \mathcal{T} , so dass $[x, y]$ komplett in der Vereinigung dieser Teilintervalle enthalten ist. Eine Überdeckung durch vier Teilintervalle ist durch die dicken Intervalle in der folgenden Abbildung skizziert.



- Seien S, F zwei Arrays der Länge n . Die Einträge der Arrays $S[i] = s_i$ und $F[i] = f_i$ beschreiben dann die n Intervalle $[s_i, f_i]$. Beschreiben Sie einen *gierigen Algorithmus* (*Greedy-Algorithmus*), der bei Eingabe (x, y, S, F, n) in Laufzeit $\mathcal{O}(n \log n)$ die kleinste Anzahl an Intervallen aus \mathcal{T} bestimmt, die notwendig ist, um das Intervall $[x, y]$ zu überdecken. Hierbei ist *kein Pseudocode* gefordert. Sie können in Ihrem Algorithmus annehmen, dass eine solche Überdeckung stets existiert.
- Analysieren Sie die Laufzeit Ihres Algorithmus.
- Beweisen Sie, dass Ihr Algorithmus eine optimale Lösung berechnet.

Name:

Matrikelnummer: