Technische Universität Wien
Institut für Computergraphik und Algorithmen
Arbeitsbereich für Algorithmen und Datenstrukturen

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

3. Übungstest WS 2007

31. Jänner 2008

Machen Sie die folgenden Angaben bitte in deutlicher Blockschrift:

Nachname: Vorname:

Matrikelnummer: Studienkennzahl:

Anzahl abgegebener Zusatzblätter:

Legen Sie bitte Ihren Studentenausweis vor sich auf das Pult.

Sie können die Lösungen entweder direkt auf die Angabeblätter oder auf Zusatzblätter schreiben, die Sie auf Wunsch von der Aufsicht erhalten. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden.

Die Verwendung von Taschenrechner, Mobiltelefonen, Skripten, Büchern, Mitschriften, Ausarbeitungen oder vergleichbaren Hilfsmitteln ist unzulässig.

Die Arbeitszeit beträgt 55 Minuten.

	A1:	A2:	A3:	Summe:
Erreichbare Punkte:	16	14	20	50
Erreichte Punkte:				

Viel Erfolg!

Aufgabe 1.A: Ω/O/Θ-Notation **(16 Punkte)**

a) (8 Punkte) Gegeben sei die folgende Funktion:

$$f(n) = \begin{cases} \log 2 \cdot n^2 - \frac{7}{2} + \frac{2n^3 \cdot \sqrt{n}}{7n}, & \text{wenn n} < 1000 \\ \frac{7n^3 \cdot \log n}{2n} + \sqrt{2} \cdot n - \frac{2}{7}, & \text{sonst.} \end{cases}$$

Kreuzen Sie in der folgenden Tabelle die zutreffenden Felder an:

$f(n)$ ist	O(.)	Ω(.)	Θ(.)	keines
$n^3 \log(n)$				
$n^2 \log(2)$				
$n^2 \sqrt{(n)}$				
$n^2 \log(n)$				

Jede Zeile wird nur dann gewertet, wenn sie vollständig richtig ist.

b) (2 Punkte) Ergänzen Sie den unten angegebenen Algorithmus so, dass er eine quadratische Laufzeit in Abhängigkeit von n in Θ-Notation besitzt.

```
m = n;
für j = 1, ..., m {
  k = [      ] ;
  solange k ≥ 0 {
    k = [      ] ;
  }
}
```

c) (2 Punkte) Schreiben Sie einen möglichst einfachen Algorithmus, der *zwei hintereinander geschaltete Schleifen* und eine Laufzeit von $\Theta(n\sqrt{n})$ in Abhängigkeit der Eingabefolge n besitzt.

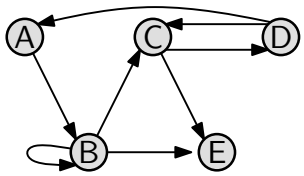
d) (4 Punkte) Gegeben sei die folgende Funktion:

$$f(n) = n^2 + \frac{3}{5} \cdot n + 12 \cdot \sqrt[3]{\frac{n}{5^3}} - 3.$$

Schätzen Sie diese Funktion *nach oben hin* möglichst scharf (genau) ab und beweisen Sie Ihre Aussage. Beachten Sie, dass zu einem Beweis auch die Angabe der notwendigen Konstanten (inkl. n_0) gehört (die Konstanten sollten relativ klein gewählt werden, müssen aber *nicht* minimal sein)!

Aufgabe 2.A: Graphen **(14 Punkte)**

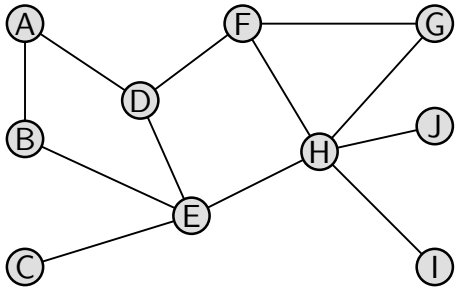
a) (4 Punkte) Gegeben sei der folgende gerichtete Graph $G(V, E)$:



Geben Sie sowohl die Adjazenzmatrix als auch die Adjazenzlistendarstellung dieses Graphen an.

b) (4 Punkte) Angenommen Sie programmieren eine Tiefensuche für sehr dünne Graphen, bei denen kein Knoten mehr als 3 Nachbarn besitzt. Welche Darstellungsform für den Graphen ist aus Effizienzgründen zu wählen: Adjazenzmatrix oder -liste? Begründen Sie Ihre Antwort, indem Sie **für beide Datenstrukturen** eine Abschätzung des Aufwands zum Finden aller Nachbarn eines Knotens $v \in V$ in Θ-Notation angeben, jeweils in Abhängigkeit einer dafür geeigneten Größe.

c) (4 Punkte) Gegeben sei der folgende ungerichtete Graph $G(V, E)$:



Auf diesem Graphen werden **Tiefen-** und **Breitensuche** durchgeführt. Welche der folgenden Listen von besuchten Knoten können dabei in genau dieser Reihenfolge in G entstehen?

Tiefensuche:	A D E C H J G F I B	□ ja	□ nein
Tiefensuche:	H I G F D A E C B J	□ ja	□ nein
Breitensuche:	E B D H C A J I G F	□ ja	□ nein
Breitensuche:	H I J G F E D A B C	□ ja	□ nein

d) (2 Punkte) Was verhindert man bei der Tiefen-/Breitensuche durch die Markierung von Knoten?

Aufgabe 3.A: Optimierung **(20 Punkte)**

a) (16 Punkte) Gegeben ist eine endliche Menge $\mathrm{M} \subset \mathbb{N}$ sowie zwei Konstanten $\in \mathbb{N}$: $c > 0$ und $n > 0$.

Schreiben Sie einen möglichst einfachen und effizienten Algorithmus (verwenden Sie z.B. begrenzte Enumeration oder dynamische Programmierung) in detailliertem Pseudocode, der eine n-elementige Auswahl $\mathrm{A} \subseteq \mathrm{M}$ von Elementen aus M trifft, sodass die Summe der Elemente in A möglichst nahe an c heran kommt:

$$\min_{\mathrm{A} \subseteq \mathrm{M}} \left| c - \sum_{i \in \mathrm{A}} i \right| \qquad \text{mit } |\mathrm{A}| = n$$

Sie können davon ausgehen, dass die Kardinalität der Menge M größer gleich n ist ($|\mathrm{M}| \geq n$). Falls es mehr als eine Auswahl gibt, die die oben erwähnten Bedingungen erfüllt, geben Sie die erste gefundene zurück.

b) (4 Punkte) Erklären sie in wenigen Worten den Unterschied zwischen *dynamischer Programmierung* und *Divide and Conquer*-Verfahren.

Technische Universität Wien
Institut für Computergraphik und Algorithmen
Arbeitsbereich für Algorithmen und Datenstrukturen

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

3. Übungstest WS 2007

31. Jänner 2008

Machen Sie die folgenden Angaben bitte in deutlicher Blockschrift:

Nachname: Vorname:

Matrikelnummer: Studienkennzahl:

Anzahl abgegebener Zusatzblätter:

Legen Sie bitte Ihren Studentenausweis vor sich auf das Pult.

Sie können die Lösungen entweder direkt auf die Angabeblätter oder auf Zusatzblätter schreiben, die Sie auf Wunsch von der Aufsicht erhalten. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden.

Die Verwendung von Taschenrechner, Mobiltelefonen, Skripten, Büchern, Mitschriften, Ausarbeitungen oder vergleichbaren Hilfsmitteln ist unzulässig.

Die Arbeitszeit beträgt 55 Minuten.

	A1:	A2:	A3:	Summe:
Erreichbare Punkte:	16	14	20	50
Erreichte Punkte:				

Viel Glück!

Aufgabe 1.B: Ω/O/Θ-Notation **(16 Punkte)**

a) (2 Punkte) Schreiben Sie einen möglichst einfachen Algorithmus, der *zwei hintereinander geschaltete Schleifen* und eine Laufzeit von $\Theta(n \log n)$ in Abhängigkeit der Eingabefolge n besitzt.

b) (2 Punkte) Ergänzen Sie den unten angegebenen Algorithmus so, dass er eine quadratische Laufzeit in Abhängigkeit von n in Θ-Notation besitzt.

```
m = [      ] ;
für j = [          ] {
  k = n;
  solange k ≥ 0 {
    k = k − 1;
  }
}
```

c) (4 Punkte) Gegeben sei die folgende Funktion:

$$f(n) = n^3 + 12 \cdot \sqrt[4]{\frac{n}{5^4}} + \frac{2}{5} \cdot n - 2.$$

Schätzen Sie diese Funktion *nach oben hin* möglichst scharf (genau) ab und beweisen Sie Ihre Aussage. Beachten Sie, dass zu einem Beweis auch die Angabe der notwendigen Konstanten (inkl. n_0) gehört (die Konstanten sollten relativ klein gewählt werden, müssen aber *nicht* minimal sein)!

d) (8 Punkte) Gegeben sei die folgende Funktion:

$$f(n) = \begin{cases} \sqrt{5} \cdot n - \frac{5}{4} + \frac{5n^3 \cdot \log n}{4n}, & \text{wenn n} > 2500 \\ \frac{4n^3 \cdot \sqrt{n}}{5n} + \log 2 \cdot n^2 - \frac{5}{4}, & \text{sonst.} \end{cases}$$

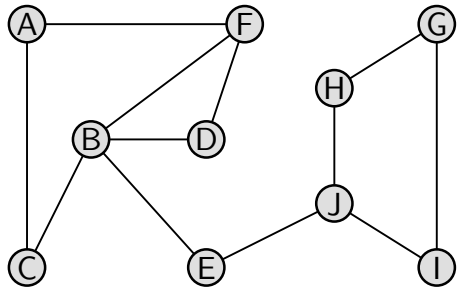
Kreuzen Sie in der folgenden Tabelle die zutreffenden Felder an:

$f(n)$ ist	Θ(.)	O(.)	Ω(.)	keines
$n^2\sqrt{(n)}$				
$n^2 \log(n)$				
$n^2 \log(2)$				
$n^3 \log(n)$				

Jede Zeile wird nur dann gewertet, wenn sie vollständig richtig ist.

Aufgabe 2.B: Graphen **(14 Punkte)**

a) (4 Punkte) Gegeben sei der folgende ungerichtete Graph $G(V, E)$:

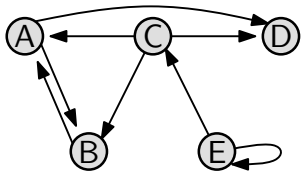


Auf diesem Graphen werden **Tiefen-** und **Breitensuche** durchgeführt. Welche der folgenden Listen von besuchten Knoten können dabei in genau dieser Reihenfolge in G entstehen?

Tiefensuche:	A F B C E D J H G I	□ ja	□ nein
Tiefensuche:	J H G I E B F A C D	□ ja	□ nein
Breitensuche:	B C F D A E J H I G	□ ja	□ nein
Breitensuche:	E B J C F D A I H G	□ ja	□ nein

b) (2 Punkte) Was verhindert man bei der Tiefen-/Breitensuche durch die Markierung von Knoten?

c) (4 Punkte) Gegeben sei der folgende gerichtete Graph $G(V, E)$:



Geben Sie sowohl die Adjazenzmatrix als auch die Adjazenzlistendarstellung dieses Graphen an.

d) (4 Punkte) Angenommen Sie programmieren eine Tiefensuche für sehr dünne Graphen, bei denen kein Knoten mehr als 4 Nachbarn besitzt. Welche Darstellungsform für den Graphen ist aus Effizienzgründen zu wählen: Adjazenzmatrix oder -liste? Begründen Sie Ihre Antwort, indem Sie **für beide Datenstrukturen** eine Abschätzung des Aufwands zum Finden aller Nachbarn eines Knotens $v \in V$ in Θ-Notation angeben, jeweils in Abhängigkeit einer dafür geeigneten Größe.

Aufgabe 3.B: Optimierung **(20 Punkte)**

a) (16 Punkte) Gegeben ist eine endliche Menge $\mathrm{M} \subset \mathbb{N}$ sowie zwei Konstanten $\in \mathbb{N}$: $c > 0$ und $n > 0$.

Schreiben Sie einen möglichst einfachen und effizienten Algorithmus (verwenden Sie z.B. begrenzte Enumeration oder dynamische Programmierung) in detailliertem Pseudocode, der eine n-elementige Auswahl $\mathrm{A} \subseteq \mathrm{M}$ von Elementen aus M trifft, sodass die Summe der Elemente in A möglichst nahe an c heran kommt:

$$\min_{\mathrm{A} \subseteq \mathrm{M}} \left| c - \sum_{i \in \mathrm{A}} i \right| \qquad \text{mit } |\mathrm{A}| = n$$

Sie können davon ausgehen, dass die Kardinalität der Menge M größer gleich n ist ($|\mathrm{M}| \geq n$). Falls es mehr als eine Auswahl gibt, die die oben erwähnten Bedingungen erfüllt, geben Sie die erste gefundene zurück.

b) (4 Punkte) Erklären sie in wenigen Worten den Unterschied zwischen *dynamischer Programmierung* und *Greedy*-Verfahren.

Technische Universität Wien
Institut für Computergraphik und Algorithmen
Arbeitsbereich für Algorithmen und Datenstrukturen

Algorithmen und Datenstrukturen 1
186.089 VO 3.0

Vorlesungsprüfung

31. Jänner 2008

Machen Sie die folgenden Angaben bitte in deutlicher Blockschrift:

Nachname: _____ Vorname: _____

Matrikelnummer: _____ Studienkennzahl: _____

Anzahl abgegebener Zusatzblätter: _____

Legen Sie während des Tests Ihren Studentenausweis vor sich auf das Pult.

Sie können die Lösungen entweder direkt auf die Angabeblätter oder auf Zusatzblätter schreiben, die Sie auf Wunsch von der Aufsicht erhalten. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu benutzen.

Die Verwendung von Taschenrechner, Mobiltelefonen, Skripten, Büchern, Mitschriften, Ausarbeitungen oder vergleichbaren Hilfsmitteln ist nicht erlaubt.

	A1:	A2:	A3:	A4:	A5:	Summe:
Erreichbare Punkte:	10	10	10	10	10	50
Erreichte Punkte:						

Viel Erfolg!

Aufgabe 1.A: Ω/O/Θ-Notation **(10 Punkte)**

a) (6 Punkte) Gegeben sei die folgende Funktion:

$$f(n) = \begin{cases} \log 2 \cdot n^2 - \frac{7}{2} + \frac{2n^3 \cdot \sqrt{n}}{7n}, & \text{wenn n} < 1000 \\ \frac{7n^3 \cdot \log n}{2n} + \sqrt{2} \cdot n - \frac{2}{7}, & \text{sonst.} \end{cases}$$

Kreuzen Sie in der folgenden Tabelle die zutreffenden Felder an:

$f(n)$ ist	O(.)	Ω(.)	Θ(.)	keines
$n^2 \log(2)$				
$n^2 \sqrt{(n)}$				
$n^2 \log(n)$				

Jede Zeile wird nur dann gewertet, wenn sie vollständig richtig ist.

b) (2 Punkte) Ergänzen Sie den unten angegebenen Algorithmus so, dass er eine quadratische Laufzeit in Abhängigkeit von n in Θ-Notation besitzt.

$m = n$;
für $j = 1, \ldots, m$ **{**
  $k =$ [ ] ;
  solange $k \geq 0$ **{**
    $k =$ [ ] ;
  }
}

c) (2 Punkte) Gegeben sei die folgende Funktion:

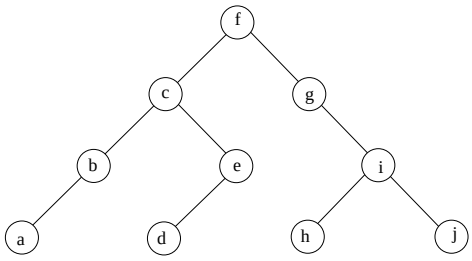
$$f(n) = n^2 + \frac{3}{5} \cdot n + 12 \cdot \sqrt[3]{\frac{n}{5^3}} - 3.$$

Schätzen Sie diese Funktion *nach oben hin* möglichst scharf (genau) ab und beweisen Sie Ihre Aussage. Beachten Sie, dass zu einem Beweis auch die Angabe der notwendigen Konstanten (inkl. n_0) gehört (die Konstanten sollten relativ klein gewählt werden, müssen aber *nicht* minimal sein)!

Aufgabe 2.A: Suchbäume **(10 Punkte)**

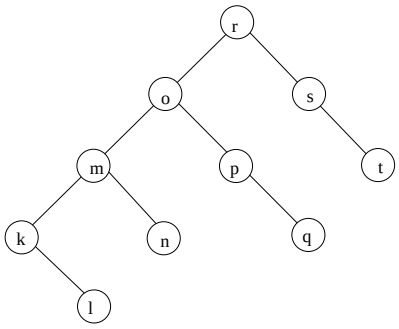
a) (4 Punkte) Welche der vier Bäume sind AVL-Bäume? Markieren Sie deutlich alle Knoten, in denen eine notwendige Bedingung verletzt ist.

A:



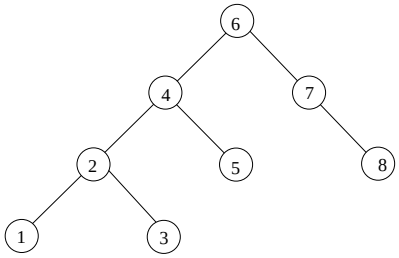
AVL-Baum: ☐ ja ☐ nein

B:



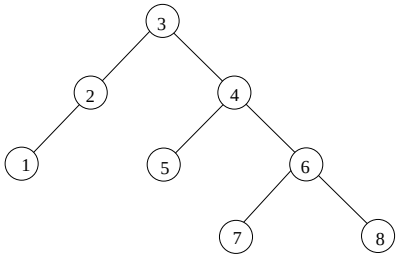
AVL-Baum: ☐ ja ☐ nein

C:



AVL-Baum: ☐ ja ☐ nein

D:



AVL-Baum: ☐ ja ☐ nein

b) (6 Punkte) Fügen Sie nacheinander die Werte

$$\langle 4, 10, 9, 13, 22, 24 \rangle$$

in einen anfangs leeren AVL-Baum ein. Skizzieren Sie alle dafür notwendigen Rotationen.

Aufgabe 3.A: Radix-Trie **(10 Punkte)**

Gehen Sie davon aus, dass T ein Radix-Trie für 5-Bit-Zahlen ist. Also hat jedes in T gespeicherte Element die Form $(b_1, b_2, b_3, b_4, b_5) \in \{0, 1\}^5$. In der Wurzel wird nach Bit b_1 unterschieden, in der zweiten Ebene nach Bit b_2 u.s.w.

a) (6 Punkte) Zeichnen Sie einen Radix-Trie, der die folgenden Elemente enthält:

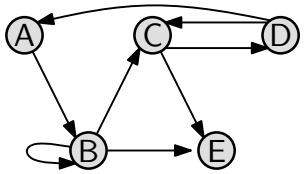
10110
11001
10101
11110
01110

b) (2 Punkte) Zeichnen Sie den gleichen Radix-Trie nach Einfügen von 01111 (Sie können sich auf den Teil des Tries beschränken, der sich verändert).

c) (2 Punkte) Geben Sie den Aufwand der Operationen *Suchen* und *Entfernen* in einem Radix Trie mit einer Schlüssellänge von l im Best Case in Θ-Notation an.

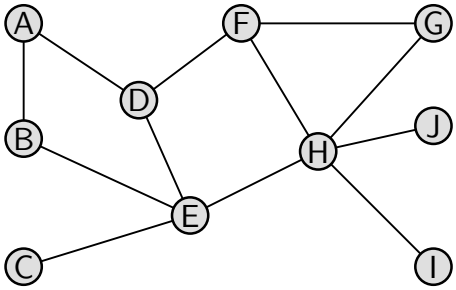
Aufgabe 4.A: Graphen **(10 Punkte)**

a) (4 Punkte) Gegeben sei der folgende gerichtete Graph $G(V, E)$:



Geben Sie sowohl die Adjazenzmatrix als auch die Adjazenzlistendarstellung dieses Graphen an.

b) (4 Punkte) Gegeben sei der folgende ungerichtete Graph $G(V, E)$:



Auf diesem Graphen werden **Tiefen-** und **Breitensuche** durchgeführt. Welche der folgenden Listen von besuchten Knoten können dabei in genau dieser Reihenfolge in G entstehen?

Tiefensuche:	A D E C H J G F I B	□ ja	□ nein
Tiefensuche:	H I G F D A E C B J	□ ja	□ nein
Breitensuche:	E B D H C A J I G F	□ ja	□ nein
Breitensuche:	H I J G F E D A B C	□ ja	□ nein

c) (2 Punkte) Was verhindert man bei der Tiefen-/Breitensuche durch die Markierung von Knoten?

Aufgabe 5.A: Optimierung **(10 Punkte)**

Gegeben ist eine endliche Menge $\mathrm{M} \subset \mathbb{N}$ sowie zwei Konstanten $\in \mathbb{N}$: $c > 0$ und $n > 0$.

Schreiben Sie einen möglichst einfachen und effizienten Algorithmus (verwenden Sie z.B. begrenzte Enumeration oder dynamische Programmierung) in detailliertem Pseudocode, der eine n-elementige Auswahl $\mathrm{A} \subseteq \mathrm{M}$ von Elementen aus M trifft, sodass die Summe der Elemente in A möglichst nahe an c heran kommt:

$$\min_{\mathrm{A} \subseteq \mathrm{M}} \left| c - \sum_{i \in \mathrm{A}} i \right| \qquad \text{mit } |\mathrm{A}| = n$$

Sie können davon ausgehen, dass die Kardinalität der Menge M größer gleich n ist ($|\mathrm{M}| \geq n$). Falls es mehr als eine Auswahl gibt, die die oben erwähnten Bedingungen erfüllt, geben Sie die erste gefundene zurück.