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

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

1. Übungstest WS 2007

16. November 2007

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:	18	16	16	50
Erreichte Punkte:				

Viel Erfolg!

Aufgabe 1.A: Ω/O/Θ-Notation **(18 Punkte)**

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

$$f(n) = \begin{cases} 2n^3 \cdot \sqrt{n} + n! + n\log_2(n), & \text{wenn } n \text{ gerade} \\ 3n^2 \log_3(n) + 2^n \cdot n + 3n^3\sqrt{n}, & \text{wenn } n \text{ ungerade.} \end{cases}$$

Kreuzen Sie in der folgenden Tabelle die zutreffenden Felder an:

$f(n)$ ist	O(.)	Ω(.)	Θ(.)	keines
$n\log_3(n)$				
$3^n \cdot n$				
$n!$				

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

b) (4 Punkte) Welche der Aussagen sind richtig? Kreuzen Sie die korrekten Aussagen an.

$\Theta(\sqrt[5]{n^2}) < \Theta(\sqrt[3]{n^2})$ ◯ ja ◯ nein $\quad n\log_3(n) = n\log_4(n)$ ◯ ja ◯ nein

$\Theta(3 \cdot \frac{n^3}{\sqrt{n}}) > \Theta(\frac{n^3}{\sqrt{n}})$ ◯ ja ◯ nein $\quad n\sqrt{n} < n\sqrt{n} + 2$ ◯ ja ◯ nein

c) (4 Punkte) Bestimmen Sie die Laufzeiten des unten angegebenen Algorithmus in Abhängigkeit von n in Θ-Notation. Verwenden Sie hierfür einen möglichst einfachen Term.

```
m = 3 · n²; l = √n;
für j = 1, ..., m {
  k = ⌊j/2⌋;
  für i = k, ..., 1 {
    solange l ≥ 1 {
      c = c + b; l = ⌊l/2⌋;
    }
  }
}
```

d) (4 Punkte) Beweisen oder widerlegen Sie folgende Aussage:
Aus $f(n) = \Omega(g(n))$ und $g(n) = O(h(n))$ folgt dass $f(n) = O(h(n))$.

Bitte wenden!

Aufgabe 2.A: Sortierverfahren **(16 Punkte)**

a) (8 Punkte) Sortieren Sie die nachfolgenden Zahlen **absteigend** mittels Fachverteilung:

$$\langle 894, 83, 632, 895, 692, 743 \rangle.$$

Geben Sie jeweils die Inhalte der einzelnen Fächer nach jeder Verteilungsphase, sowie das Feld nach Ende jeder Sammelphase an.

b) (8 Punkte) Analysieren Sie den gegebene Sortieralgorithmus **Gnomesort**.

Gnomesort$(A[1, \ldots, n])$

```
i = 1;
key;
solange i ≤ n {
  falls (i == 1) oder (A[i − 1] ≤ A[i]) dann {
    i = i + 1;
  } sonst {
    key = A[i];
    A[i] = A[i − 1];
    A[i − 1] = key;
    i = i − 1;
  }
}
```

- Überlegen Sie sich die Funktionsweise anhand der Beispielfolge $\langle 7, 3, 1, 5 \rangle$. Sortieren Sie diese mit **Gnomesort**, schreiben Sie dabei den Zustand der Eingabefolge zu Beginn **jedes** Durchlaufs der `solange`-Schleife auf. Beschreiben Sie anschließend mit wenigen Worten die Funktionsweise des Sortieralgorithmus. Welche Bedeutung hat dabei die Laufvariable i?
- Bestimmen Sie die Laufzeit von **Gnomesort** in Θ-Notation in Abhängigkeit der Eingabegröße n.

Aufgabe 3.A: Theorie **(16 Punkte)**

a) (2 Punkte) Nennen Sie zwei Voraussetzungen, die für lineare Sortierverfahren erfüllt sein müssen.

b) (4 Punkte) Beschreiben Sie das Prinzip **Teile und Erobere** (divide and conquer) kurz. Was ist der Vorteil dieses Prinzips? Nennen Sie ein Sortierverfahren aus der Vorlesung, das nach diesem Prinzip funktioniert.

c) (2 Punkte) Sie möchten eine Zahlenfolge sortieren, die höchstwahrscheinlich schon (fast) korrekt sortiert ist. Welches Sortierverfahren aus der Vorlesung würden Sie hier einsetzen, beziehungsweise welches Verfahren ist für diesen Fall nicht gut geeignet?

d) (6 Punkte) In dem gegebenen Feld ist die Heapbedingung für einen **Minimum-Heap** verletzt.

$$\langle 3, 9, 25, 5, 10, 10, 7, 6, 10, 24, 13, 30, 27, 35, 28 \rangle$$

- Stellen Sie das Feld als binären Baum dar und markieren Sie eindeutig die Knoten, die die Heapeigenschaft verletzen.
- Stellen Sie die Heapeigenschaft für einen Minimum-Heap her. Zeichnen Sie den korrekten Heap als binären Baum und beschreiben Sie die Vorgehensweise entweder in kurzen Worten oder stellen Sie die durchgeführten Operationen im Baum grafisch dar.

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

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

1. Übungstest WS 2007

16. November 2007

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:	18	16	16	50
Erreichte Punkte:				

Viel Glück!

Aufgabe 1.B: Ω/O/Θ-Notation **(18 Punkte)**

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

$$f(n) = \begin{cases} 3n^2\sqrt{n} + 2^n \cdot n + 3n^3 \log(n), & \text{wenn } n \text{ gerade.} \\ 2\log(n) \cdot \sqrt{n} + n! + n\sqrt{n}, & \text{wenn } n \text{ ungerade} \end{cases}$$

Kreuzen Sie in der folgenden Tabelle die zutreffenden Felder an:

$f(n)$ ist	$\Theta(.)$	$\Omega(.)$	O(.)	keines
$n!$				
$n \log_3(n)$				
$3^n \cdot n$				

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

b) (4 Punkte) Welche der Aussagen sind richtig? Kreuzen Sie an.

$\Theta(\frac{n^3}{\sqrt{n}}) < \Theta(2 \cdot \frac{n^3}{\sqrt{n}})$ ◯ ja ◯ nein $\quad \Theta(\sqrt[3]{n^2}) > \Theta(\sqrt[5]{n^2})$ ◯ ja ◯ nein

$n^2 \log_2(n) = n^2 \log_3(n)$ ◯ ja ◯ nein $\quad n^3/n < n^3/n + 2$ ◯ ja ◯ nein

c) (4 Punkte) Bestimmen Sie die Laufzeiten des unten angegebenen Algorithmus in Abhängigkeit von n in Θ-Notation. Verwenden Sie hierfür einen möglichst einfachen Term.

```
b = √n; a = 3 · n^3;
für j = 1, ..., a {
  c = ⌊j/2⌋;
  für i = c, ..., 1 {
    solange b ≥ 1 {
      l = l + m; b = ⌊b/2⌋;
    }
  }
}
```

d) (4 Punkte) Beweisen oder widerlegen Sie folgende Aussage:
Aus $f(n) = O(g(n))$ und $g(n) = \Omega(h(n))$ folgt dass $f(n) = \Omega(h(n))$.

Bitte wenden!

Aufgabe 2.B: Theorie **(16 Punkte)**

a) (6 Punkte) In dem gegebenen Feld ist die Heapbedingung für einen **Maximum-Heap** verletzt.

$$\langle 25, 7, 13, 10, 9, 10, 24, 1, 3, 8, 7, 6, 5, 10, 17 \rangle$$

- Stellen Sie das Feld als binären Baum dar und markieren Sie eindeutig die Knoten, die die Heapeigenschaft verletzen.
- Stellen Sie die Heapeigenschaft für einen Maximum-Heap her. Zeichnen Sie den korrekten Heap als binären Baum und beschreiben Sie die Vorgehensweise entweder in kurzen Worten oder stellen Sie die durchgeführten Operationen im Baum grafisch dar.

b) (2 Punkte) Sie möchten eine Zahlenfolge sortieren, die höchstwahrscheinlich schon (fast) korrekt sortiert ist. Welches Sortierverfahren aus der Vorlesung würden Sie hier einsetzen, beziehungsweise welches Verfahren ist für diesen Fall nicht gut geeignet?

c) (4 Punkte) Beschreiben Sie das Prinzip **Teile und Erobere** (divide and conquer) kurz. Was ist der Vorteil dieses Prinzips? Nennen Sie ein Sortierverfahren aus der Vorlesung, das nach diesem Prinzip funktioniert.

d) (4 Punkte) Nennen Sie zwei Voraussetzungen, die für lineare Sortierverfahren erfüllt sein müssen.

Aufgabe 3.B: Sortierverfahren **(16 Punkte)**

a) (8 Punkte) Analysieren Sie den gegebene Sortieralgorithmus **Giantsort**.

Giantsort$(A[1,\ldots,n])$
  $i = 1$;
  key;
  solange $i \leq n$ **{**
    falls $(i > 1)$ **und** $(A[i-1] > A[i])$ **dann {**
      key $= A[i]$;
      $A[i] = A[i-1]$;
      $A[i-1] =$ key;
      $i = i - 1$;
    } sonst {
      $i = i + 1$;
    }
  }

- Überlegen Sie sich die Funktionsweise anhand der Beispielfolge $\langle 8, 6, 2, 4\rangle$. Sortieren Sie diese mit **Giantsort**, schreiben Sie dabei den Zustand der Eingabefolge zu Beginn **jedes** Durchlaufs der `solange`-Schleife auf. Beschreiben Sie anschließend mit wenigen Worten die Funktionsweise des Sortieralgorithmus. Welche Bedeutung hat dabei die Laufvariable i?
- Bestimmen Sie die Laufzeit von **Giantsort** in Θ-Notation in Abhängigkeit der Eingabegröße n.

b) (8 Punkte) Sortieren Sie die nachfolgenden Zahlen **absteigend** mittels Fachverteilung:

$$\langle 643, 829, 29, 793, 163, 143\rangle.$$

Geben Sie jeweils die Inhalte der einzelnen Fächer nach jeder Verteilungsphase, sowie das Feld nach Ende jeder Sammelphase an.

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

Algorithmen und Datenstrukturen 1
186.089 VO 3.0

Vorlesungsprüfung

16. November 2007

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:

$$f(n) = \begin{cases} 2n^3 \cdot \sqrt{n} + n! + n\log_2(n), & \text{wenn } n \text{ gerade} \\ 3n^2\log_3(n) + 2^n \cdot n + 3n^3\sqrt{n}, & \text{wenn } n \text{ ungerade.} \end{cases}$$

Kreuzen Sie in der folgenden Tabelle die zutreffenden Felder an:

$f(n)$ ist	O(.)	Ω(.)	Θ(.)	keines
$n\log_3(n)$				
$3^n \cdot n$				
$n!$				

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

b) (2 Punkte) Welche der Aussagen sind richtig? Kreuzen Sie die korrekten Aussagen an.

$\Theta(\sqrt[5]{n^2}) < \Theta(\sqrt[3]{n^2})$ ◯ ja ◯ nein $\quad n\log_3(n) = n\log_4(n)$ ◯ ja ◯ nein

$\Theta(3 \cdot \frac{n^3}{\sqrt{n}}) > \Theta(\frac{n^3}{\sqrt{n}})$ ◯ ja ◯ nein $\quad n\sqrt{n} < n\sqrt{n} + 2$ ◯ ja ◯ nein

c) (2 Punkte) Bestimmen Sie die Laufzeiten des unten angegebenen Algorithmus in Abhängigkeit von n in Θ-Notation. Verwenden Sie hierfür einen möglichst einfachen Term.

```
m = 3 · n²; l = √n;
für j = 1, ..., m {
  k = ⌊j/2⌋;
  für i = k, ..., 1 {
    solange l ≥ 1 {
      c = c + b; l = ⌊l/2⌋;
    }
  }
}
```

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

a) (5 Punkte) Sortieren Sie die nachfolgenden Zahlen **absteigend** mittels Fachverteilung:

$$\langle 894, 83, 632, 895, 692, 743 \rangle.$$

Geben Sie jeweils die Inhalte der einzelnen Fächer nach jeder Verteilungsphase, sowie das Feld nach Ende jeder Sammelphase an.

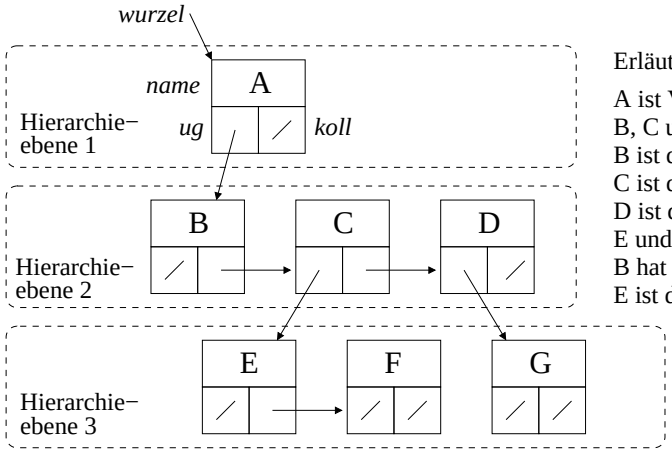
b) (5 Punkte) Analysieren Sie den gegebene Sortieralgorithmus **Gnomesort**.

Gnomesort$(A[1, \ldots, n])$
  $i = 1$;
  key;
  solange $i \leq n$ **{**
    falls $(i == 1)$ **oder** $(A[i-1] \leq A[i])$ **dann {**
      $i = i + 1$;
    } sonst {
      key $= A[i]$;
      $A[i] = A[i-1]$;
      $A[i-1] =$ key;
      $i = i - 1$;
    }
  }

- Überlegen Sie sich die Funktionsweise anhand der Beispielfolge $\langle 7, 3, 1, 5 \rangle$. Sortieren Sie diese mit **Gnomesort**, schreiben Sie dabei den Zustand der Eingabefolge zu Beginn **jedes** Durchlaufs der `solange`-Schleife auf. Beschreiben Sie anschließend mit wenigen Worten die Funktionsweise des Sortieralgorithmus. Welche Bedeutung hat dabei die Laufvariable i?
- Bestimmen Sie die Laufzeit von **Gnomesort** in Θ-Notation in Abhängigkeit der Eingabegröße n.

Aufgabe 3.A: Bäume **(10 Punkte)**

a) (8 Punkte) Gegeben sei ein Organigramm einer Firma in Form eines binären Baumes. Jeder Knoten repräsentiert einen Mitarbeiter, welcher durch seinen Namen *name* eindeutig identifiziert wird (Namensgleichheiten werden hier ausgeschlossen). Der Wurzelknoten repräsentiert den Firmenleiter. Jeder Mitarbeiter kann beliebig viele direkte Untergebene haben. Dazu wird als linker Nachkomme *ug* immer der erste direkte Untergebene gespeichert. Als rechter Nachkomme *koll* wird der nächste Kollege (= nächster Mitarbeiter auf derselben Hierarchieebene, der denselben direkten Vorgesetzten hat) gespeichert. Die folgende Skizze verdeutlicht die Baumstruktur:



Erläuterungen:

A ist Vorgesetzter von B, C und D
B, C und D sind direkte Untergebene von A
B ist der erste direkte Untergebene von A
C ist der nächste Kollege von B
D ist der nächste Kollege von C
E und F sind Kollegen (Untergebene von C)
B hat keine direkten Untergebenen
E ist der erste direkte Untergebene von C

Ein Knoten enthält also folgende Informationen:

name:	Name des Mitarbeiters als String
ug:	Verweis auf ersten direkten Untergebenen
koll:	Verweis auf nächsten Kollegen

Schreiben Sie unter Verwendung der gegebenen Datenstruktur einen effizienten Algorithmus Subordinates(m, num) in Pseudocode. Dieser soll für den Baum, auf dessen Wurzel m verweist, die Namen all jener Mitarbeiter ausgeben, die weniger als num direkte Untergebene haben.

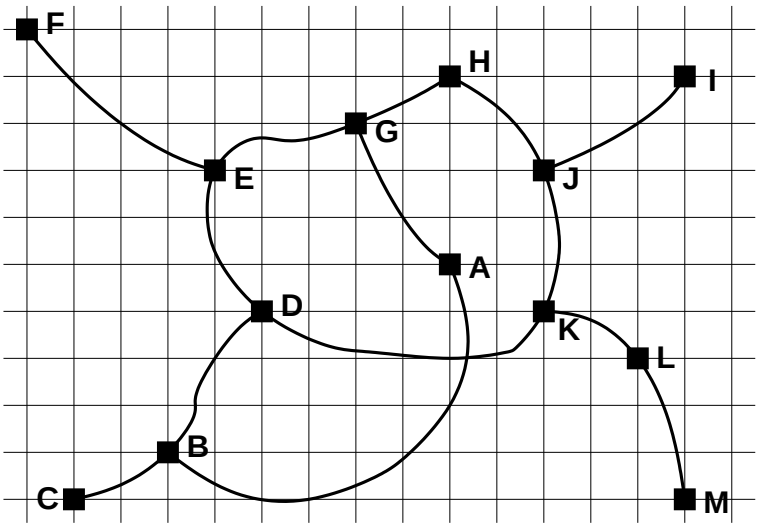
b) (2 Punkte)
Geben Sie den Aufwand Ihres Algorithmus in Θ-Notation an und begründen Sie Ihre Antwort.

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

- (5 Punkte)

  Auf dem unten abgebildeten Graph $G(V, E)$ wird **Tiefensuche** durchgeführt. Welche der folgenden Listen von besuchten Knoten können in genau dieser Reihenfolge bei einer Tiefensuche auf G entstehen?

  a) F E D B C K L M J I H G A ◯ ja ◯ nein
  b) E G H J K D B C A L M I F ◯ ja ◯ nein
  c) D K L M J H G E F A B C I ◯ ja ◯ nein
  d) J I H G A B D C E F K L M ◯ ja ◯ nein



- (5 Punkte)

  Nun wird **Breitensuche** auf dem obigen Graphen angewendet. Diese wird mit Hilfe der Datenstruktur *Queue* realisiert. Folgende Operationen sind auf einer Queue definiert:

  – `enq(X)`: Fügt ein Objekt `X` in die Queue ein.
  – `deq()`: Entfernt das älteste Objekt aus der Queue und liefert es zurück.

  Starten Sie in Knoten A die Breitensuche, die Nachbarschaftsfunktion liefert Knoten in lexikographischer Ordnung. Schreiben Sie nun **die ersten 15** Queue-Operationen in genau der Reihenfolge auf, in der sie bei der Breitensuche in G auftreten.

  1: `enq(A)` (Initialisierung)
  2: `deq() {= A}` (Start der Breitensuche)
  3: `enq(B)`
  4: `...`

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

Beantworten Sie folgende Punkte in möglichst wenigen, aber genauen Worten.

a) (3 Punkte)
Beschreiben Sie die Spannbaum-Heuristik für das symmetrische Traveling Salesman Problem.

b) (3 Punkte) Ein Sortierverfahren heißt *stabil*, wenn sich bei Schlüsseln mit gleichen Werten die Reihenfolge der Schlüssel relativ zueinander nach der Sortierung nicht verändert.

- Beschreiben Sie ein Szenario, in welchem man auf ein stabiles Sortierverfahren angewiesen ist.
- Nennen Sie je ein Beispiel für ein stabiles und ein nicht stabiles Sortierverfahren, das in der Vorlesung behandelt wurde.

c) (2 Punkte) Nennen Sie zwei Voraussetzungen, die für lineare Sortierverfahren erfüllt sein müssen.

d) (2 Punkte)
Was bedeutet es, wenn Sie für ein Minimierungsproblem einen ε-*approximativen Algorithmus* haben?

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

Algorithmen und Datenstrukturen 1
186.089 VO 3.0

Vorlesungsprüfung

16. November 2007

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.B: Ω/O/Θ-Notation **(10 Punkte)**

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

$$f(n) = \begin{cases} 3n^2\sqrt{n} + 2^n \cdot n + 3n^3 \log(n), & \text{wenn } n \text{ gerade.} \\ 2\log(n) \cdot \sqrt{n} + n! + n\sqrt{n}, & \text{wenn } n \text{ ungerade} \end{cases}$$

Kreuzen Sie in der folgenden Tabelle die zutreffenden Felder an:

$f(n)$ ist	$\Theta(.)$	$\Omega(.)$	$O(.)$	keines
$n!$				
$n \log_3(n)$				
$3^n \cdot n$				

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

b) (2 Punkte) Welche der Aussagen sind richtig? Kreuzen Sie an.

$\Theta(\frac{n^3}{\sqrt{n}}) < \Theta(2 \cdot \frac{n^3}{\sqrt{n}})$ ◯ ja ◯ nein $\quad \Theta(\sqrt[3]{n^2}) > \Theta(\sqrt[5]{n^2})$ ◯ ja ◯ nein

$n^2 \log_2(n) = n^2 \log_3(n)$ ◯ ja ◯ nein $\quad n^3/n < n^3/n + 2$ ◯ ja ◯ nein

c) (2 Punkte) Bestimmen Sie die Laufzeiten des unten angegebenen Algorithmus in Abhängigkeit von n in Θ-Notation. Verwenden Sie hierfür einen möglichst einfachen Term.

```
b = √n; a = 3 · n^3;
für j = 1, ..., a {
  c = ⌊j/2⌋;
  für i = c, ..., 1 {
    solange b ≥ 1 {
      l = l + m; b = ⌊b/2⌋;
    }
  }
}
```

Aufgabe 2.B: Sortierverfahren **(10 Punkte)**

a) (5 Punkte) Analysieren Sie den gegebene Sortieralgorithmus **Giantsort**.

```
Giantsort(A[1, ..., n])
  i = 1;
  key;
  solange i ≤ n {
    falls (i > 1) und (A[i − 1] > A[i]) dann {
      key = A[i];
      A[i] = A[i − 1];
      A[i − 1] = key;
      i = i − 1;
    } sonst {
      i = i + 1;
    }
  }
```

- Überlegen Sie sich die Funktionsweise anhand der Beispielfolge $\langle 8, 6, 2, 4\rangle$. Sortieren Sie diese mit **Giantsort**, schreiben Sie dabei den Zustand der Eingabefolge zu Beginn **jedes** Durchlaufs der `solange`-Schleife auf. Beschreiben Sie anschließend mit wenigen Worten die Funktionsweise des Sortieralgorithmus. Welche Bedeutung hat dabei die Laufvariable i?
- Bestimmen Sie die Laufzeit von **Giantsort** in Θ-Notation in Abhängigkeit der Eingabegröße n.

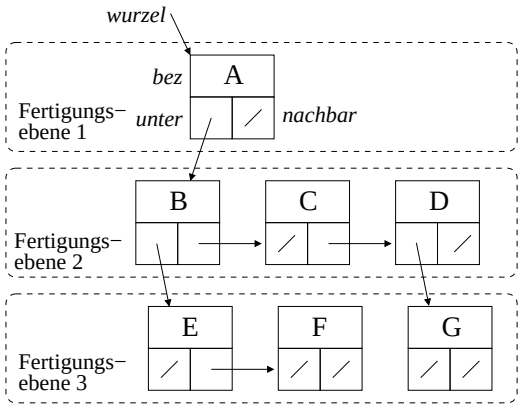
b) (5 Punkte) Sortieren Sie die nachfolgenden Zahlen **absteigend** mittels Fachverteilung:

$$\langle 643, 829, 29, 793, 163, 143\rangle.$$

Geben Sie jeweils die Inhalte der einzelnen Fächer nach jeder Verteilungsphase, sowie das Feld nach Ende jeder Sammelphase an.

Aufgabe 3.B: Bäume **(10 Punkte)**

a) (8 Punkte)
Gegeben sei eine hierarchische Stückliste eines Produktes in Form eines binären Baumes. Jeder Knoten repräsentiert eine Komponente, welche durch ihre Bezeichnung *bez* eindeutig identifiziert wird (Namensgleichheiten werden hier ausgeschlossen). Der Wurzelknoten repräsentiert das Fertigprodukt. Jede Komponente kann aus beliebig vielen Unterkomponenten bestehen. Dazu wird als linker Nachkomme *unter* immer die erste direkte Unterkomponente gespeichert. Als rechter Nachkomme *nachbar* wird die nächste Nachbarkomponente (= nächste Komponente auf derselben Fertigungsebene, die in derselben übergeordneten Komponente enthalten ist) gespeichert. Die folgende Skizze verdeutlicht die Baumstruktur:



Erläuterungen:

A enthält die Unterkomponenten B, C und D
B, C und D sind direkte Unterkomponenten von A
B ist die erste direkte Unterkomponente von A
C ist die nächste Nachbarkomponente von B
D ist die nächste Nachbarkomponente von C
E und F sind Nachbarkomponenten (sind in B enthalten)
C hat keine direkten Unterkomponenten
E ist die erste direkte Unterkomponente von B

Ein Knoten enthält also folgende Informationen:

bez:	Produktbezeichnung als String
unter:	Verweis auf erste direkte Unterkomponente
nachbar:	Verweis auf nächste Nachbarkomponente

Schreiben Sie unter Verwendung der gegebenen Datenstruktur einen effizienten Algorithmus Komponenten(k, *zaehl*) in Pseudocode. Dieser soll für den Baum, auf dessen Wurzel k verweist, die Namen all jener Komponenten ausgeben, die aus weniger als *zaehl* direkten Unterkomponenten bestehen.

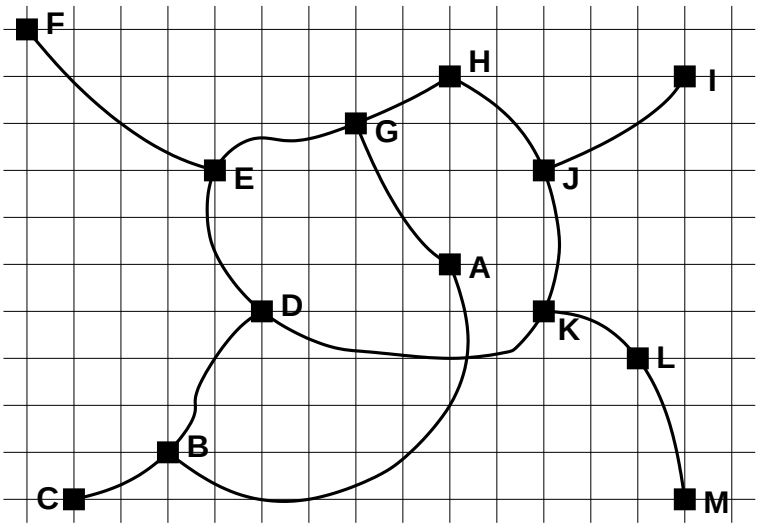
b) (2 Punkte)
Geben Sie den Aufwand Ihres Algorithmus in Θ-Notation an und begründen Sie Ihre Antwort.

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

- (5 Punkte)

  Auf dem unten abgebildeten Graph $G(V, E)$ wird **Breitensuche** durchgeführt. Welche der folgenden Listen von besuchten Knoten können in genau dieser Reihenfolge bei einer Breitensuche auf G entstehen?

  a) F E D G B K A H C L J M I ◯ ja ◯ nein
  b) E F D G B K A H C L M J I ◯ ja ◯ nein
  c) D K L M J H G E F A B C I ◯ ja ◯ nein
  d) J I H K G D L A E B M F C ◯ ja ◯ nein



- (5 Punkte)

  Nun wird **Tiefensuche** auf dem obigen Graphen angewendet. Diese wird mit Hilfe der Datenstruktur *Stack* (statt mit einer Rekursion) realisiert. Folgende Operationen sind auf einem Stack definiert:

  – `push(X)`: Legt ein Objekt X auf den Stack.
  – `pop()`: Entfernt das oberste Objekt vom Stack und liefert es zurück.

  Starten Sie in Knoten A die Tiefensuche, die Nachbarschaftsfunktion liefert Knoten in lexikographischer Ordnung. Schreiben Sie nun **die ersten 15** Stack-Operationen in genau der Reihenfolge auf, in der sie bei der Tiefensuche in G auftreten.

  1: `push(A)` (Initialisierung)
  2: `push(B)` (Start der Tiefensuche)
  3: `...`

Aufgabe 5.B: Theorie **(10 Punkte)**

Beantworten Sie folgende Punkte in möglichst wenigen, aber genauen Worten.

a) (2 Punkte)
Welche drei Funktionen stellt eine *Union-Find*-Datenstruktur zu Verfügung? (Beschreiben Sie jede Funktion in einem kurzen Satz.)

b) (4 Punkte) Beschreiben Sie das Prinzip **Teile und Erobere** (divide and conquer) kurz. Was ist der Vorteil dieses Prinzips? Nennen Sie ein Sortierverfahren aus der Vorlesung, das nach diesem Prinzip funktioniert.

c) (2 Punkte)
Was bedeutet es, wenn Sie für ein Minimierungsproblem einen *ε-approximativen Algorithmus* haben?

d) (2 Punkte) Sie möchten eine Zahlenfolge sortieren, die höchstwahrscheinlich schon (fast) korrekt sortiert ist. Welches Sortierverfahren aus der Vorlesung würden Sie hier einsetzen, beziehungsweise welches Verfahren ist für diesen Fall nicht gut geeignet?