Algorithmen und Programmieren 3 **Matrikelnummer:** ______________
WS 2003/2004, Zwischenklausur **NAME:** ______________________

Abgabe bis Dienstag, 16. Dezember 2003

Anleitung: Streichen Sie *eine* Aufgabe deutlich auf dem Angabeblatt. Diese Aufgabe wird nicht in die Bewertung einbezogen. Wenn Sie selbst keine Aufgabe streichen, wird die erste Aufgabe nicht in die Bewertung einbezogen. Bearbeiten Sie die übrigen drei Aufgaben.

Schreiben Sie jede Aufgabe auf ein getrenntes Blatt. Jede Aufgabe hat 10 Punkte.

1. Ein binärer Suchbaum soll Einträge speichern, die außer dem Schlüssel x_i noch ein „Gewicht" w_i enthalten. Die Algorithmen zum Suchen, Einfügen und Löschen in binären Suchbäumen sollen so erweitert werden, dass man jederzeit zu einem Wert a das Gesamtgewicht aller Einträge finden kann, deren Schlüssel $x < a$ ist.

   Die Laufzeit für alle Operationen soll höchstens proportional zur Höhe des Baumes sein.

   Beschreiben Sie die Felder, die in den Knoten enthalten sind (zusammen mit einer Erklärung der Bedeutung, sofern sie nicht offensichtlich ist), und ein Programmstück (eine oder mehrere Methoden in Java oder Funktionen in Haskell), das die Intervallabfrage durchführt. (Einfügen und Löschen müssen Sie *nicht* programmieren.)

2. Programmieren Sie eine *stabile* Variante von Sortieren durch Verschmelzen (*mergesort*) in Java oder Haskell. Erläutern Sie die Stellen im Programm, die für die Stabilität wichtig sind.

3. Sortieren durch Zählen:
   Das folgende Programmstück soll n verschiedene `double`-Zahlen sortieren, die in einem Array `a[0]`, ..., `a[n-1]` gespeichert sind, mit Hilfe eines Arrays `pos[i]`, das die Anzahl der Elemente zählt, die kleiner als `a[i]` sind.

```
int [ ] pos = new int[n];
for (int i=0; i<n; i++) { // Zähle Elemente kleiner als a[i]:
    pos[i] = 0;
    for (int j=0; j<n; j++)
        if (a[j]<a[i]) pos[i]++;
    // pos[i] ist die Position von a[i] in der sortierten Reihenfolge.
}
// In die richtige Reihenfolge umordnen:
for (int i=0; i<n; i++) a[pos[i]] = a[i];
```

   (a) Was ist der Speicherbedarf und die asymptotische Laufzeit dieses Programmes?
   (b) Warum funktioniert dieses Programm nicht richtig? Geben Sie ein Beispiel an, bei dem das Array `a[0]`, ..., `a[n-1]` am Ende *nicht aufsteigend sortiert* ist.
   (c) Stellen Sie das Programm richtig, sodass am Ende die sortierten Zahlen in `a[0]`, ..., `a[n-1]` stehen, ohne dass die die asymptotische Laufzeitschranke in Aufgabe (a) überschritten wird.
   (d) Was passiert, wenn die Eingabe gleiche Werte enthält?
   Erstellen Sie eine Variante des Programms, die auch solche Eingaben sortiert.

4. Beweisen Sie: Ein binärer Baum mit n Blättern hat mindestens $n-1$ innere Knoten. (Innere Knoten haben 1 oder 2 Kinder, und Blätter haben keine Kinder.) Sie dürfen nicht einfach die Aussage von Übung 31 über volle binäre Bäume verwenden, aber Sie dürfen die Lösung (den Beweis) dieser Aufgabe anpassen.

Algorithmen und Programmieren 3 **Matrikelnummer:** ____________
WS 2003/2004, Zwischenklausur **FAMILIENNAME:** ________________
Dienstag, 20. Januar 2004. **Vorname:** ________________

Anleitung: Streichen Sie *eine* Aufgabe deutlich auf dem Angabeblatt. Diese Aufgabe wird nicht in die Bewertung einbezogen. Wenn Sie selbst keine Aufgabe streichen, wird die erste Aufgabe nicht in die Bewertung einbezogen. Bearbeiten Sie die übrigen drei Aufgaben.

Bearbeiten Sie jede Aufgabe auf einem getrennten Blatt. Jede Aufgabe hat 10 Punkte. Bearbeitungszeit: 90 Minuten

1. Man möchte einen binären Baum wie bei einer Halde als *array* speichern: Die Kinder des Knotens `a[i]` stehen an den Stellen `a[2*i]` und `a[2*i+1]`. Nehmen Sie an, dass die Bäume (im Gegensatz zu einer Halde) nicht ausgeglichen sein müssen. Bestimmen Sie die notwendige Größe des Arrays zum Speichern von n Elementen in Abhängigkeit von n,

   (a) wenn der Baum die Höhe $2\lceil \log n \rceil$ hat,
   (b) im schlimmsten Fall,
   (c) wenn die Höhe des Baumes nur um 2 größer als die kleinstmögliche Höhe ist.

   Geben Sie außerdem an, in welchem dieser Fälle man mit $O(n)$ Speicher auskommt.

2. Eliminieren Sie die Endrekursion aus dem folgenden Programmstück, das die Anzahl der übereinstimmenden Elemente in zwei Vektoren `a[0]`, ..., `a[n-1]` und `b[0]`, ..., `b[n-1]` zählt. Gehen Sie dabei systematisch Schritt für Schritt vor. Schreiben Sie eine einzige Methode als Ersatz für `anz`.

```
int anz(int[ ] a, int[ ] b, int n) { return anz2(a,b,n,0); }
int anz2(int[ ] a, int[ ] b, int n, int s)
{ if(n==0) return s;
  if(a[n-1]==b[n-1]) return anz2(a,b,n-1,s+1);
  return anz2(a,b,n-1,s);
}
```

3. Sortieren: Das folgende Programmstück soll n verschiedene `int`-Zahlen zwischen 1 und m sortieren, die in einem Array `a[0]`, ..., `a[n-1]` gespeichert sind.

```
int [ ] count = new int[m+1];
for (int j=1; j<=m; j++) count[j]=0;
for (int i=0; i<n;  i++) count[a[i]]++;
int i=0;
for (int j=1; j<=m; j++)
     for (int k=0; k<count[j]; k++)  {a[i]=j; i++;}
```

   (a) Bestimmen Sie die asymptotische Laufzeit dieses Programms.
   (b) Warum muss man in der letzten Zeile in der Anweisung `a[i]=j` keinen Überlauf des Indexes i über die Feldgrenzen von `a` befürchten? Geben Sie eine ausführliche Begründung.
   (c) Funktioniert dieses Verfahren auch, wenn die Zahlen nicht verschieden sind? Was muss man gegebenenfalls ändern, damit es für diesen Fall funktioniert?

4. Schreiben Sie ein Programm in Java oder Haskell, das für einen gegebenen binären Suchbaum die *mittlere innere Weglänge* berechnet; sie ist um 1 kleiner als die erwartete Anzahl von Vergleichen, die zum Finden eines zufällig ausgewählten Knotens in dem Baum im Durchschnitt erforderlich sind. (Jeder Knoten wird mit gleicher Wahrscheinlichkeit gesucht.)

Algorithmen und Programmieren 3 **WS 2003/2004, Endklausur**
Donnerstag, 19. Februar 2004

Anleitung: Streichen Sie *eine* Aufgabe deutlich auf dem Angabeblatt. Diese Aufgabe wird nicht in die Bewertung einbezogen. *Alle Antworten sind zu begründen!* Jede Aufgabe hat 10 Punkte. Bearbeitungszeit: 90 Minuten

1. Die folgende Java-Klasse implementiert (teilweise) die Mengenoperationen für Mengen, die Teilmengen von $\{1, 2, \ldots, 100\}$ sind.

```
class Menge100
{  boolean [ ] a = new boolean[100];
   int größe = 0;
   public void enfügen(int x)
   {  if(a[x-1]==false) { a[x-1]=true; größe++; }
   }
   public boolean istleer() { return größe==0; }
   ...
}
```

   Geben Sie die Abstraktionsfunktion und die Darstellungsinvariante an, und beweisen Sie, dass die Operationen `einfügen` und `istleer` (mit geeigneten Vorbedingungen, falls nötig) korrekt implementiert sind.

2. Eine Zeichenkette $p_1 p_2 \ldots p_m$ heißt *periodisch* mit Periode k, $(1 \le k \le m)$ wenn $p_i = p_{i+k}$ für $1 \le i < i + k \le m$ ist, mit anderen Worten, wenn

$$p_1 \ldots p_{m-k} = p_{k+1} \ldots p_m$$

   ist. (Für $k = m$ ist die Folge unperiodisch.)

   (a) (3 Punkte) Bestimmen Sie die Periode und berechnen sie die Verschiebefunktion h für das Muster *ababcababcabab*. Verwenden Sie die Definition der Verschiebefunktion aus der Vorlesung.[1] Berechnen Sie auch den Wert $h(m+1)$.

   (b) (0 Punkte) Berechnen sie die Verschiebefunktion für *abcabcabcab*.

   (c) (7 Punkte) Wie kann man mit Hilfe der Verschiebefunktion die (kleinste) Periode k einer Zeichenkette bestimmen?

3. Konstruieren Sie mit dem Huffman-Algorithmus einen optimalen Kode für die Verteilung $(p_1, \ldots, p_6) = (\frac{8}{25}, \frac{2}{25}, \frac{1}{25}, \frac{5}{25}, \frac{5}{25}, \frac{4}{25})$. Zeigen Sie in einzelnen Schritten, wie der Algorithmus abläuft, und zeichnen Sie den optimalen Kode-Baum.
   Ist der optimale Kode eindeutig?

4. Leiten Sie aus der algebraischen Spezifikation für Mengen

$$\begin{aligned}
istenthalten(x, leer) &= falsch && (1)\\
istenthalten(x, einfüge(x, M)) &= wahr && (2)\\
istenthalten(x, einfüge(y, M)) &= istenthalten(x, M), \quad \text{für } x \neq y && (3)\\
istenthalten(x, lösche(x, M)) &= falsch && (4)\\
istenthalten(x, lösche(y, M)) &= istenthalten(x, M), \quad \text{für } x \neq y && (5)
\end{aligned}$$

   folgende Identität her:

$$istenthalten(u, einfüge(x, lösche(u, M))) = istenthalten(u, einfüge(x, leer))$$

   Geben Sie dabei in jedem Beweisschritt die Nummer der verwendeten Gleichung an.

---

[1] Die ursprüngliche Definition der Verschiebefunktion aus der Vorlesung lautet

$$h_i = \max \{\, k \mid 1 \le k < i,\ p_1 \ldots p_{k-1} = p_{i-k+1} \ldots p_{i-1} \,\} \cup \{0\}, \qquad \text{für } i \ge 1.$$

Die verbesserte Verschiebefunktion vom 12. Übungsblatt ist so definiert:

$$h_i = \max \{\, k \mid 1 \le k < i,\ p_1 \ldots p_{k-1} = p_{i-k+1} \ldots p_{i-1} \text{ und } p_k \neq p_i \,\} \cup \{0\}, \qquad \text{für } i \ge 1.$$

Algorithmen und Programmieren 3
erste Nachklausur zur ersten Teilklausur
Donnerstag, 26. Februar 2004.

Anleitung: Streichen Sie *eine* Aufgabe deutlich auf dem Angabeblatt. Diese Aufgabe wird nicht in die Bewertung einbezogen. Wenn Sie selbst keine Aufgabe streichen, wird die erste Aufgabe nicht in die Bewertung einbezogen. Bearbeiten Sie die übrigen drei Aufgaben.

Bearbeiten Sie jede Aufgabe auf einem getrennten Blatt. Jede Aufgabe hat 10 Punkte. Bearbeitungszeit: 90 Minuten

---

1. Wir wollen eine Variante von a-b-Bäumen konstruieren, bei der weniger Speicher verschwendet wird. Es sei $a = 50$. Wie klein kann man b wählen, wenn man

   - beim Einfügen *zwei* Geschwisterknoten zu Hilfe nimmt, falls diese den Überlauf aufnehmen können, und
   - beim Entfernen bis zu *drei* Geschwisterknoten zum Ausborgen in Betracht zieht?

   Begründen Sie Ihre Antwort!

2. Gegeben ist ein Graph mit den Knoten $V = \{1, 2, \ldots, n\}$ und m Kanten in Adjazenzlistenspeicherung. Die Kanten in (i, j) den Adjazenzlisten sollen nach den Endknoten umsortiert werden; das heißt, in der Adjazenzliste von i soll die Kante (i, j) vor (i, k) kommen, wenn $j < k$ ist.

   (a) Skizzieren Sie einen Algorithmus, der diese Sortieraufgabe *in linearer Zeit*, das heißt, in $O(m + n)$ Zeit löst.

   (b) Schreiben Sie ein Java-Programm für diese Aufgabe.

3. Eliminieren Sie die Endrekursion aus dem folgenden Programmstück, das eine verkettete Liste umkehrt. Gehen Sie dabei systematisch Schritt für Schritt vor, und geben Sie alle Zwischenschritte bei der Transformation explizit an. Schreiben Sie am Ende eine einzige Methode als Ersatz für `umkehre`.

```
class Liste { Object wert; Liste next; }

static Liste umkehre(Liste x)
{ if(x==null) return x;
  else return umkehre3(null, x, x.next);
}
static Liste umkehre3(Liste davor, Liste x, Liste danach)
{ x.next = davor;
  if (danach==null) return x;
  else return umkehre3(x, danach, danach.next);
}
```

4. Schreiben Sie ein Java-Programm, das die Knoten eines binären Baumes in *Niveau-Reihenfolge* ausgibt, das heißt, zuerst die Wurzel, dann die Knoten mit Tiefe 1, dann mit Tiefe 2, und so weiter.

   Ihr Programm soll lineare Laufzeit haben.

Algorithmen und Programmieren 3 **Matrikelnummer:** ____________
Nachklausur zur Endklausur **FAMILIENNAME:** ________________
WS 2003/2004. Dienstag, 6. April 2004. **Vorname:** ________________

Anleitung: Streichen Sie *eine* Aufgabe deutlich auf dem Angabeblatt. Diese Aufgabe wird nicht in die Bewertung einbezogen. Wenn Sie selbst keine Aufgabe streichen, wird die erste Aufgabe nicht in die Bewertung einbezogen. Bearbeiten Sie die übrigen drei Aufgaben.

Bearbeiten Sie jede Aufgabe auf einem getrennten Blatt. Jede Aufgabe hat 10 Punkte. Bearbeitungszeit: 90 Minuten

Aufgabe	1	2	3	4	Summe
Punkte	10	10	10	10	30
Punkte					

---

1. Die sogenannte *Erdős-Zahl*[1] ist nach dem berühmten ungarischen Mathematiker Paul Erdős (1913–1996) benannt. Er selbst hat Erdős-Zahl 0. Alle seine Koautoren, mit denen er wissenschaftliche Arbeiten publiziert hat, haben Erdős-Zahl 1. Wer eine gemeinsame Arbeit mit jemandem mit Erdős-Zahl 1 geschrieben hat, aber nicht mit Erdős selbst, hat Erdős-Zahl 2, und so weiter. Personen, die nicht auf diese Weise mit Erdős verbunden sind, haben Erdős-Zahl ∞.

   Der *Kollaborationsgraph* enthält alle lebenden oder toten Personen als Knoten. Zwei Knoten sind durch eine Kante verbunden, wenn die beiden Personen gemeinsame Autoren einer wissenschaftlichen Publikation sind.

   (a) Geben Sie einen *Algorithmus* an, der für eine gegebene Person im Kollaborationsgraphen die Erdős-Zahl berechnet.

   (b) Geben Sie einen Algorithmus an, der die größte endliche Erdős-Zahl bestimmt.

   Algorithmen aus der Vorlesung können Sie direkt verwenden; die müssen Sie nicht "ausprogrammieren".

2. (a) (5 Punkte) Zeichne sie den

      i. digitalen Suchbaum

      ii. den komprimierten digitalen Suchbaum

   für die Bitketten `0011000101`, `001111001`, `01011`, `00111111`, `01010`, `001101`. Zeigen Sie, wie die Bäume nach dem Einfügen des Wertes `0011000` aussehen. Entfernen Sie `001111001` und zeichnen Sie die Bäume danach.

   (b) (5 Punkte) Wie kann man aus der Verschiebefunktion des Wortes $x\#w\$$ berechnen, ob x ein Teilwort von w ist? (Hier sind $\#$ und $\$$ neue Buchstaben, die nicht in x und w vorkommen.) *Begründen Sie Ihre Antwort!*

[1] `http://personalwebs.oakland.edu/~grossman/erdoshp.html`

3. Eine stückweise lineare Funktionen $f\colon (u,v] \to \mathbb{R}$ ist durch eine zusammenhängende Folge von Intervallen mit entsprechenden linearen Funktionen $f(x) = ax + b$ für jedes Intervall gegeben, z. B.

$$f(x) = \begin{cases} -x, & \text{für } -1 < x \le 0 \\ 2x, & \text{für } 0 < x \le 2 \\ -3x + 9, & \text{für } 2 < x \le 4 \end{cases}$$

Die Datenstruktur soll folgende Operationen erlauben:

(a) Erzeugen einer Funktion $f(x) = ax + b$ mit einem einzigen Intervall $(u, v]$.

(b) Berechnung des Wertes einer solche Funktion $f(x)$ an einer gegebenen Stelle x.

(c) Addition zweier stückweise linearer Funktionen.

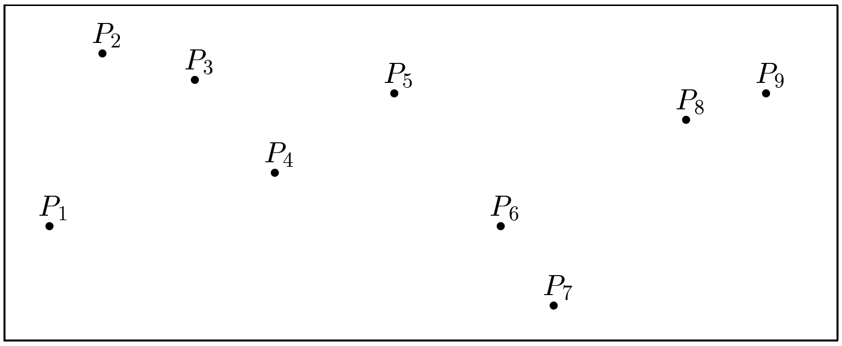
Nehmen Sie der Einfachheit halber an, dass alle Intervalle halboffen sind, und zwar links offen und rechts abgeschlossen, wie im obigen Beispiel.

Der Datentyp ist durch die folgenden Axiome (mit geeigneten Signaturen) algebraisch spezifiziert:

$$\begin{aligned} \mathit{berechne}(\mathit{stLinFun}(u,v,a,b),x) &= \begin{cases} ax + b, & \text{für } u \le x < v \\ 0, & \text{sonst} \end{cases} \\ \mathit{berechne}(\mathit{add}(f,g),x) &= \mathit{berechne}(f,x) + \mathit{berechne}(g,x) \end{aligned}$$

Implementieren Sie diesen Datentyp in Java oder Haskell und beweisen Sie die Korrektheit.

4. Bestimmen Sie die untere konvexe Hülle der folgenden sortierten Punktmenge $P_1, \ldots, P_9$ mit dem inkrementellen Algorithmus aus der Vorlesung. Geben Sie die Folge der Orientierungstests an, die der Algorithmus ausführt.



Algorithmen und Programmieren 3, WS 2003/2004.
zweite Nachklausur zur ersten Teilklausur
Dienstag, 6. April 2004.

Anleitung: Streichen Sie *eine* Aufgabe deutlich auf dem Angabeblatt. Diese Aufgabe wird nicht in die Bewertung einbezogen. Wenn Sie selbst keine Aufgabe streichen, wird die erste Aufgabe nicht in die Bewertung einbezogen. Bearbeiten Sie die übrigen drei Aufgaben.

Bearbeiten Sie jede Aufgabe auf einem getrennten Blatt. Jede Aufgabe hat 10 Punkte. Bearbeitungszeit: 90 Minuten

---

1. *Rot-Schwarz-Bäume* sind binäre Bäume mit folgenden Eigenschaften:[1]

   (a) Jeder innere Knoten hat zwei Kinder.

   (b) Jeder Knoten ist entweder als *rot* oder als *schwarz* gekennzeichnet.

   (c) Die Wurzel und alle Blätter sind schwarz.

   (d) Die Kinder eines roten Knotens sind schwarz.

   (e) Ein schwarzer Knoten kann höchstens ein rotes Kind haben.

   (f) Alle Wege von der Wurzel zu den Blättern enthalten die gleiche Anzahl von schwarzen Knoten.

   Die *schwarze Höhe* ist die um eins verminderte Anzahl der schwarzen Knoten auf jedem Weg von der Wurzel zu einem Blatt.

   (a) Welche Höhe kann ein Baum mit schwarzer Höhe h' mindestens und höchstens haben?

   (b) Wie viele rote Knoten kann ein Baum mit schwarzer Höhe h' mindestens und höchstens haben?

   (c) Wie viele schwarze Knoten kann ein Baum mit schwarzer Höhe h' mindestens und höchstens haben?

   Begründen Sie Ihre Antworten.

2. Fügen Sie nacheinander die Werte 5, 3, 9, 2, 7, 10, 11, 4, 1 in einen anfangs leeren 2-3-Baum ein. Entfernen Sie dann die Element 5 und 7. Zeichnen Sie den 2-3-Baum nach jeder Operation auf.

3. Für eine gegebene Folge von n verschiedenen Zahlen $x_1, x_2, \ldots, x_n$ soll die Anzahl der *Inversionen* (Fehlstände) bestimmt werden, das sind die Paare (x_i, x_j) mit $1 \leq i < j \leq n$ und $x_i > x_j$. Zum Beispiel hat die Folge $(3, 5, 10, 1, 6)$ vier Inversionen, nämlich die Paare $(3, 1)$, $(5, 1)$, $(10, 1)$ und $(10, 6)$.

   Zeigen Sie, wie man den Algorithmus *Sortieren durch Verschmelzen* so erweitern kann, dass er in $O(n \log n)$ Zeit die Anzahl der Inversionen berechnet. Geben Sie insbesondere genau an, was man beim Verschmelzen machen muss.

   Wahlweise dürfen Sie auch einen anderen Algorithmus angeben, der das Problem in $O(n \log n)$ Zeit löst.

4. Schreiben Sie ein Programm in Java oder Haskell, das eine sortierte Liste von n ganzen Zahlen in einen binären Suchbaum der kleinstmöglichen Höhe $h = \lceil \log_2(n+1) \rceil - 1$ umwandelt.

   (Ihr Programm muss *nicht* vollständig in dem Sinn sein, dass Sie zum Beispiel das Suchen und alle notwendigen und nützlichen Methoden für eine Klasse `Baum` programmieren müssen. Beschränken Sie sich auf die gestellte Aufgabe.)

---

[1] Diese Definition stimmt mit der Definition auf dem 6. Übungsblatt überein.