Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Institut für Theoretische Informatik
Peter Widmayer
Michel Gatto

Prüfung
Datenstrukturen und Algorithmen
D-INFK

7. März 2007

Name, Vorname: _____

Stud.-Nummer: _____

Ich bestätige mit meiner Unterschrift, dass ich diese Prüfung unter regulären Bedingungen ablegen konnte und dass ich die untenstehenden Hinweise gelesen und verstanden habe.

Unterschrift: _____

Hinweise:

- Ausser einem Wörterbuch dürfen Sie keine Hilfsmittel verwenden.
- Bitte schreiben Sie Ihre StudentInnen-Nummer auf **jedes** Blatt.
- Melden Sie sich bitte **sofort**, wenn Sie sich während der Prüfung in irgendeiner Weise bei der Arbeit gestört fühlen.
- Bitte verwenden Sie für jede Aufgabe ein neues Blatt. Pro Aufgabe kann nur eine Lösung angegeben werden. Ungültige Lösungsversuche müssen klar durchgestrichen werden.
- Bitte schreiben Sie **lesbar** mit blauer oder schwarzer Tinte. Wir werden nur bewerten, was wir lesen können.
- Die Prüfung dauert 120 Minuten. **Keine Angst!** Wir rechnen nicht damit, dass irgendjemand alles löst! Sie brauchen bei weitem nicht alle Punkte, um die Bestnote zu erreichen.

Viel Erfolg!

Stud.-Nummer: _____

Aufgabe	1	2	3	4	5	**Σ**
Mögl. Punkte	9	7	10	10	10	45
Punkte						

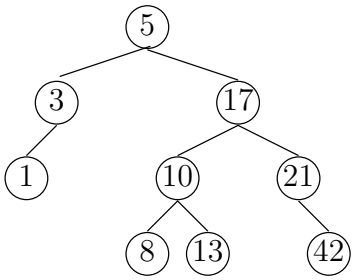
Aufgabe 1:

Hinweise:

1. In dieser Aufgabe sollen Sie **nur die Ergebnisse** angeben. Diese können Sie direkt bei den Aufgaben notieren.
2. Sofern Sie die Notationen, Algorithmen und Datenstrukturen aus der Vorlesung "Datenstrukturen & Algorithmen" verwenden, sind Erklärungen oder Begründungen nicht notwendig. Falls Sie jedoch andere Methoden benutzen, müssen Sie diese **kurz** soweit erklären, dass Ihre Ergebnisse verständlich und nachvollziehbar sind.
3. Als Ordnung verwenden wir für Buchstaben die alphabetische Reihenfolge, für Zahlen die aufsteigende Anordnung gemäss ihrer Grösse.

1 P a) In der folgenden Tabelle ist ein Min-Heap in der üblichen Form gespeichert.
$[1, 3, 4, 6, 5, 8, 10, 7, 9, 10, 12, 13, 9]$
Wie sieht die Tabelle aus, nachdem das kleinste Element entfernt wurde und die Heap-Bedingung wieder hergestellt wurde?

1 P b) Fügen Sie den Schlüssel 7 in den AVL-Baum ein und balancieren Sie ihn wieder falls nötig.



1 P c) Im unten stehenden Array sehen Sie die Ausgangslage für "natural merge sort". Führen Sie eine Iteration durch (einen Durchlauf durch das ganze Array).

12	41	29	37	12	22	5	10	12	14	17	47	4	51

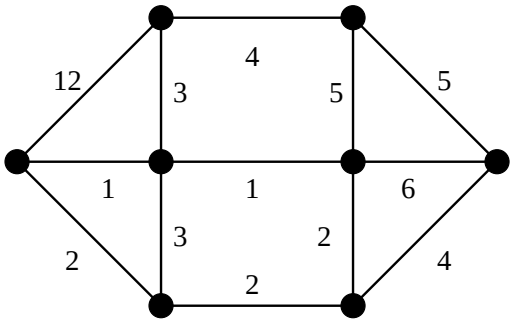
1 P d) Fügen Sie die Schlüssel $20, 3, 18, 10, 41$ in dieser Reihenfolge mittels Double Hashing in die Hashtabelle ein.

Die zu verwendenden Hash-Funktionen sind $h(k) = (k \mod 11)$ und $h'(k) = 1 + (k \mod 9)$.

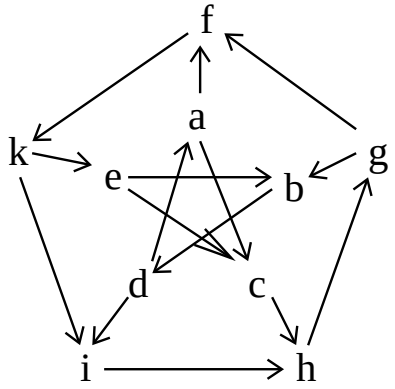
	23	02			49			19	31	05
0	1	2	3	4	5	6	7	8	9	10

1 P e) Zeichnen Sie den binären Suchbaum, dessen Post-Order-Traversierung die Folge $5, 7, 9, 11, 10, 8, 13, 21, 20, 22, 14, 12$ ergibt.

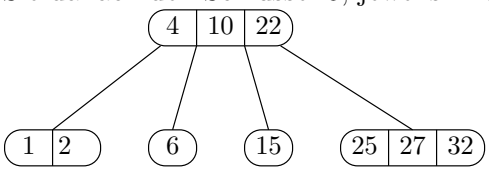
1 P f) Markieren Sie im untenstehenden gewichteten Graphen die Kanten eines minimalen Spannbaums.



1 P g) Der folgende gerichtete Graph wird mit Tiefensuche traversiert. Die Suche startet beim Knoten a. Geben Sie eine Reihenfolge an, in der die Knoten erreicht werden können.



1 P h) Fügen Sie den Schlüssel 29 in den unten stehenden B-Baum der Ordnung 4 ein und löschen Sie danach den Schlüssel 6, jeweils mit den zugehörigen Strukturänderungen.



Nach Einfügen von 29:	Nach Löschen von 6:

1 P i) Gegeben sind acht Schlüssel mit der relativen Anzahl der Zugriffe. Erstellen Sie mit Hilfe des Huffman-Algorithmus einen optimalen Codierungsbaum (Trie).

Schlüssel	a	b	c	d	e	f	g	h
Anzahl Zugriffe	5	24	14	6	25	6	14	6

Aufgabe 2:

1 P a) Geben Sie für die untenstehenden Funktionen eine **Reihenfolge** an, so dass Folgendes gilt: Wenn Funktion f links von Funktion g steht, so gilt $f \in O(g)$.

Beispiel: Die drei Funktionen n^3, n^7, n^9 sind bereits in der entsprechenden Reihenfolge, da $n^3 \in O(n^7)$ und $n^7 \in O(n^9)$ gilt.

- n^n
- $n \log n$
- $\sum_{i=0}^{n} i$
- $\sum_{i=0}^{n} \frac{1}{2^i}$
- $\frac{n}{\log n}$
- $n\sqrt{n}$
- $\Pi_{i=1}^{n} i$

3 P b) Gegeben ist die folgende Rekursionsgleichung:

$$T(n) := \begin{cases} 2T(\frac{n}{2}) + 10n + 10 & n > 1 \\ 0 & n = 1 \end{cases}$$

Geben Sie eine geschlossene (d.h. nicht-rekursive) Formel für $T(n)$ an und beweisen Sie diese.

Hinweise:
(1) Sie können annehmen, dass n eine Potenz von 2 ist.
(2) Für $q \neq 1$ gilt: $\sum_{i=0}^{k} q^i = \frac{q^{k+1}-1}{q-1}$.

1 P

c) Geben Sie die asymptotische Laufzeit in Abhängigkeit von n für folgenden Algorithmus in Theta-Notation an:

```
from i:= n+5 until i < 1 loop
   from j := 1 until j > i loop
      j := j + 1
   end
   i := i // 7
end
```

1 P

d) Geben Sie die asymptotische Laufzeit in Abhängigkeit von n für folgenden Algorithmus in Theta-Notation an:

```
k := 0
from i:= 1 until i > n loop
   k := k + i
   i := i + 1
end
from i := n until i < 1 loop
   k := k + i
   i := i // 2
end
```

1 P

e) Geben Sie die asymptotische Laufzeit in Abhängigkeit von n für folgenden Algorithmus in Theta-Notation an:

```
from i:= 8 until i > n // 2 loop
   from j:=i until j < 2 loop
      j:= j // 3 + 1;
   end
   i:= i + 1
end
```

Aufgabe 3:
Nach einer aufwändigen Prüfungssession entscheiden Sie, einen Skiurlaub in einem Skigebiet zu machen, das eine grosse Gesamt-Pisten-Länge (in km) vorzuweisen hat. Sie haben ein Verzeichnis von Skigebieten mit jeweils zugehöriger Pistenkilometerzahl und möchten nun für eine beliebige, ganze Zahl k wissen, wie viele Skigebiete mit mindestens k Pistenkilometern es im Verzeichnis gibt. Sie möchten eine solche "Skigebiets-Anzahl-Anfrage" für verschiedene Anfrageparameter k wiederholt rechnergestützt durchführen und überlegen daher, welche Datenstruktur diese Anfrage gut unterstützt.

Die Daten sind in einem Array `resorts: ARRAY [ RESORT ]` gespeichert. Die Klasse `RESORT` ist wie folgt definiert.

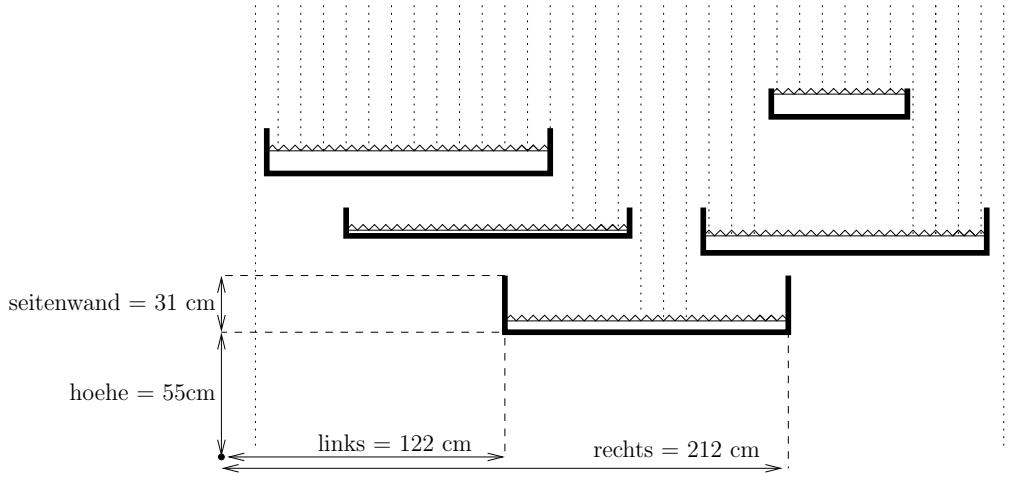
```
class RESORT
feature -- Access
  name: STRING
        -- Name des Skigebiets
  length: INTEGER
        -- Pistenlaenge (in km)
end -- class RESORT
```

3 P a) Beschreiben Sie eine Datenstruktur für eine feste Menge von Skigebieten, welche eine Skigebiets-Anzahl-Anfrage möglichst effizient unterstützt. Sie dürfen zu diesem Zweck auch bekannte Datenstrukturen als Grundlage benutzen und geeignet erweitern.

3 P b) Beschreiben Sie einen Algorithmus zur Durchführung einer solchen Anfrage auf Ihrer Datenstruktur aus Teilaufgabe a). Geben Sie sowohl die Idee des Algorithmus als auch eine Eiffel-ähnliche Implementierung in Pseudocode an. Welche Laufzeit erreichen Sie?

4 P c) Machen wir das Problem etwas schwieriger: Die Menge der verzeichneten Skigebiete samt Pistenlängenangabe soll sich dynamisch ändern können, indem Skigebiete hinzugefügt und entfernt werden dürfen. Schlagen Sie eine Datenstruktur vor, mit der sowohl Anfragen als auch dynamische Änderungen möglichst effizient realisiert werden. Beschreiben Sie die Datenstruktur sowie die dynamische Anpassung derselben möglichst genau und vollständig. Machen Sie eine möglichst genaue Aussage zur Laufzeit aller Operationen.

Aufgabe 4:
Als "Kunst am Bau" sind aussen an der Fassade des geplanten Informatikneubaus Kübel in einem zweidimensionalen Arrangement folgender Art vorgesehen:



Die Kübel haben verschiedene Breite, Seitenwand-Höhe und Position an der Aussenwand, sind aber alle gleich tief (in der dritten Dimension, auf dieser Skizze nicht zu sehen). Alle Massangaben sind in Zentimetern (cm) wie auf der Skizze für einen Kübel angegeben.
Jeder Kübel ist wie in der Klasse KUEBEL spezifiziert, und alle Kübel sind im Array `behaelter: ARRAY [ KUEBEL ]` gespeichert.

```
class KUEBEL
feature -- ACCESS
   links: INTEGER
    -- Abstand des linken Endes des Kuebels vom linken Gebaeude-Ende, in cm
   rechts: INTEGER
    -- Abstand des rechten Endes des Kuebels vom linken Gebaeude-Ende, in cm
   hoehe: INTEGER
    -- Abstand des Kuebels vom Boden, in cm
   seitenwand: INTEGER
    -- Hoehe der Seitenwand des Kuebels, in cm
end -- class KUEBEL
```

Gelegentlich regnet es, und da kein Wind bläst, fällt der Regen senkrecht von oben nach unten, wie in der Skizze zu sehen.

3 P a) Es regnet eine Menge von $r \in \mathbb{N}$ Kubikzentimetern Wasser pro Quadratzentimeter Grundfläche. Beschreiben Sie kurz in Worten einen Algorithmus, der bestimmt, ob dadurch einer der Kübel überläuft (wenn zu Beginn alle Kübel leer sind).

1 P b) Geben Sie die Laufzeit Ihres Algorithmus aus Teilaufgabe a) in Abhängigkeit von der Zahl der Kübel an.

3 P c) Schreiben Sie Ihre Lösung für Teilaufgabe a) in Pseudocode.

3 P d) Nun wollen Sie bestimmen, welche Niederschlagsmenge für Ihr Kübelarrangement (bei anfangs leeren Kübeln) nötig ist, damit ein Kübel überläuft, und welcher Kübel das ist (oder: welche Kübel das sind, falls es mehrere sind). Beschreiben Sie in Worten einen Algorithmus dafür und geben Sie seine Laufzeit an.

Aufgabe 5:
In einer Bar möchte die Besitzerin eine Wand neu tapezieren. Ihr stehen verschiedene, farbige Tapetenblätter T zur Verfügung. Die Blätter sind gleich hoch (vom Boden bis zur Decke), haben jedoch verschiedene Breiten $b_i, i = 1, \ldots, n$ (ganzzahlig, in cm) und einen ganzzahligen subjektiven Schönheitswert $s_i > 0$. Die Besitzerin möchte nun die Blätter so auswählen, dass die Auswahl die Wand der Länge L (in cm) genau abdeckt, und dass die ausgewählten Blätter eine grösstmögliche Summe der Schönheitswerte erreichen. Genaues Abdecken der Wand bedeutet natürlich, dass die Summe der Breiten der gewählten Blätter exakt L ergibt.

Beispiel: Die Wand ist 500 cm lang, und es stehen Blätter mit (Breite, Schönheit) von (100,50), (150,60), (350,60), (250,15) zur Verfügung. Wir wählen das erste, das zweite und das vierte Blatt in dieser Aufzählung.

2 P a) Erstellen Sie ein rekursives Programm in Pseudocode, welches den maximalen Schönheitswert für eine genau passende Auswahl an Blättern berechnet (und Null liefert, falls es keine solche Auswahl gibt). Geben Sie die Laufzeit Ihres Algorithmus an.

5 P b) Entwerfen Sie einen Algorithmus nach dem Muster der dynamischen Programmierung, der diesen maximalen Schönheitswert berechnet. Geben Sie die Laufzeit Ihres Algorithmus an.

1 P c) Beschreiben Sie in Worten, wie durch Rückverfolgung in der Lösungstabelle gemäss b) die Blätter-Auswahl für die Wand gefunden werden kann. Geben Sie die Laufzeit für die Rückverfolgung an.

2 P d) Entwerfen Sie einen Algorithmus nach dem Muster der dynamischen Programmierung, der eine möglichst schöne Auswahl von Blättern berechnet für den Fall, dass man nicht nur eine Wand bedecken will, sondern k Wände eines Zimmers. Jede Wand einzeln muss dabei genau passend abgedeckt werden. Geben Sie die Laufzeit Ihres Algorithmus an.