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
Beat Gfeller

Prüfung
Datenstrukturen und Algorithmen
D-INFK

6. Februar 2008

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: _____

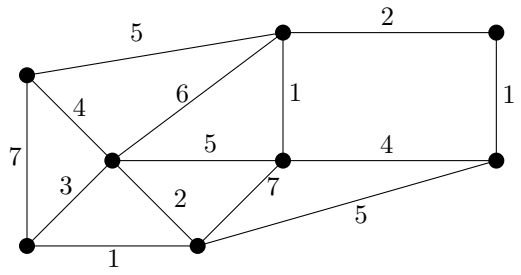
Aufgabe	1	2	3	4	5	**Σ**
Mögl. Punkte	9	7	10	9	8	43
Punkte						

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.

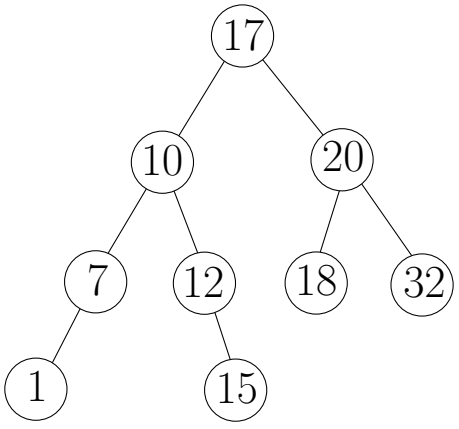
1 P a) Zeichnen Sie in den folgenden Graphen einen minimalen Spannbaum ein:



1 P b) Das untenstehende Array enthält die Elemente eines Min-Heaps in der üblichen Form gespeichert. Wie sieht das Array aus, nachdem das Minimum entfernt wurde und die Heap-Bedingung wieder hergestellt wurde?

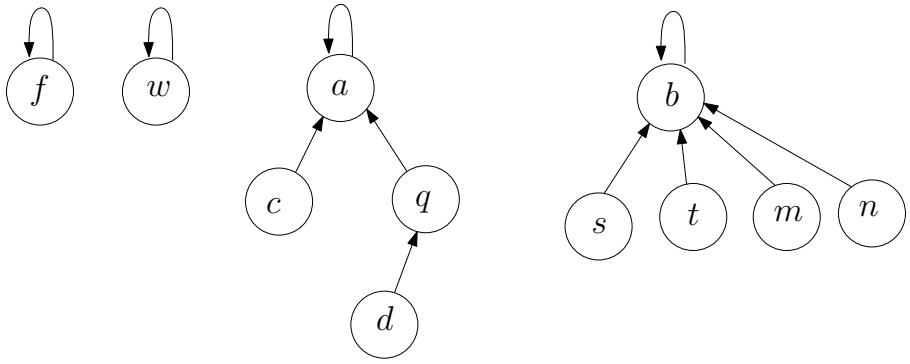
3	5	9	22	11	14	13	32	50	20

1 P c) Geben Sie die Preorder-Reihenfolge der Knoten im folgenden Baum an:



1 P

d) Führen Sie auf der folgenden Union-Find-Datenstruktur nacheinander folgende Operationen aus: Union(a, b), Union(Find$(s), w)$. Benutzen Sie dazu das Verfahren "Vereinigung nach Höhe" (auch genannt "kleiner-in-grösser"), und zeichnen Sie die nach diesen zwei Operationen resultierende Datenstruktur.



1 P

e) Fügen Sie die Schlüssel $20, 3, 18, 29, 41$ in dieser Reihenfolge mittels Offenem Hashing in die Hashtabelle ein, und benutzen Sie dabei Quadratisches Sondieren. Die zu verwendende Hash-Funktion ist $h(k) = (k \mod 11)$.

	23	17			49	5		19	31	
0	1	2	3	4	5	6	7	8	9	10

1 P

f) Geben Sie eine kürzestmögliche Folge von Zugriffen an, welche, ausgehend von der Liste

$$\text{A} \rightarrow \text{D} \rightarrow \text{C} \rightarrow \text{Q} \rightarrow \text{Z} \rightarrow \text{B}$$

mittels der Move-To-Front-Regel folgende Liste ergibt:

$$\text{A} \rightarrow \text{B} \rightarrow \text{C} \rightarrow \text{D} \rightarrow \text{Q} \rightarrow \text{Z}$$

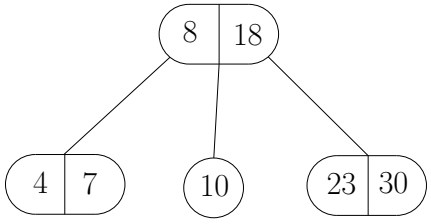
1 P

g) Führen Sie auf dem gegebenen Array einen Aufteilungsschritt des Sortieralgorithmus Quicksort durch. Benutzen Sie als Pivot das am rechten Ende stehende Element im Array.

45	89	17	59	3	5	18	22	90	13	72	6	39

1 P

h) Gegeben sei der folgende B-Baum der Ordnung 3. Fügen Sie in diesen zuerst den Schlüssel 6 ein, und löschen Sie danach den Schlüssel 10 aus dem B-Baum.



1 P

i) Zeichnen Sie den binären Suchbaum, dessen Postorder-Traversierung die Folge
1, 6, 7, 5, 11, 12, 20, 15, 10, 8
ergibt.

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.

- $\frac{n}{\log n}$
- 2^{1024}
- $3^{\sqrt{n}}$
- $\sum_{i=1}^{n} \frac{i}{2}$
- $2^{3 \log_2 n}$
- $n\sqrt{n}$
- $\log(n^{17})$

3 P b) Gegeben ist die folgende Rekursionsgleichung:

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

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 until i = n // 2 loop
   from j := 1 until j = i loop
      j := j + 1
   end
   i := i - 1
end
```

1 P

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

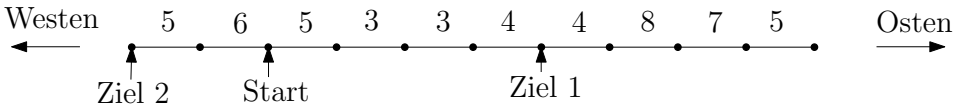
```
from k := 1 until k > n loop
   k := 2*k
   from i := 1 until i > k loop
      i := 2*i
   end
end
```

1 P

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

```
from i := 0 until i > n loop
  from j := 1 until j*j > n loop
     j := 2*j
  end
  i := i + j
end
```

Aufgabe 3:
Nach der anstrengenden Prüfungssession entscheiden Sie, zur Erholung eine Wanderung in den Bergen zu machen.
Das Wandergebiet haben Sie schon festgelegt, aber noch nicht entschieden, welche Strecke Sie wandern möchten. Das Gebiet besteht aus einer Sequenz von aufeinanderfolgenden Orten:



Dank Ihres Wanderführers kennen Sie die Distanz (in Kilometern) zwischen jedem benachbarten Paar von Orten (A, B). Die Gesamtdistanz einer Wanderung ist dann die Summe der Distanzen ihrer Teilstrecken. Beachten Sie, dass man während einer Wanderung nie die Richtung wechselt, also entweder stets nach Osten oder stets nach Westen wandert.
Im Beispiel auf dem Bild wäre die Gesamtdistanz für die Wanderung vom Start zum Ziel 1 also $5 + 3 + 3 + 4 = 15$, zum Ziel 2 wäre die Gesamtdistanz $6 + 5 = 11$.

Die Daten sind in einem Array `orte: ARRAY[ORT]` gespeichert. In diesem Array liegen die Orte nach ihrer Lage sortiert vor. Die Klasse `ORT` ist wie folgt definiert.

```
class ORT
feature -- Access
  name: STRING
        -- Name des Ortes
  length: INTEGER
        -- Distanz von diesem Ort zum nächsten Ort in Richtung Osten
        -- (beim östlichsten Ort ist dieser Wert undefiniert)
end -- class ORT
```

3 P a) Für eine feste Sequenz von Orten mit Distanz-Angaben für jede Teilstrecke möchten Sie nun wiederholt folgende Art von Anfragen effizient beantworten können: Gegeben der Startort (anhand seines Indexes im Array `orte`), sowie eine gewünschte Maximal-Distanz $D_{\max}$, finde einen Zielort, so dass die Gesamtdistanz der Wanderung möglichst nahe bei $D_{\max}$ liegt, aber nicht grösser ist. Sie dürfen dazu die Daten in einem ersten Schritt, der nicht mehr als $O($`orte.count`$)$ Zeit brauchen soll, geeignet vorverarbeiten.

Beschreiben Sie eine Datenstruktur, sowie gegebenenfalls die Vorverarbeitung, welche solche Anfragen möglichst effizient unterstützt.

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 Implementierung in Eiffel an. Welche Laufzeit erreichen Sie?

4 P c) Machen wir das Problem etwas schwieriger: Nun sei der Startort der Wanderung nicht mehr vorgegeben. Zudem nehmen wir an, dass die Daten *nicht* vorverarbeitet werden dürfen (gegeben ist nur das Array `orte`). Eine Anfrage soll also, gegeben eine Maximal-Distanz $D_{\max}$, ein Paar (Startort, Zielort) liefern, so dass die Distanz möglichst nahe bei $D_{\max}$ liegt, aber nicht grösser ist. Beschreiben Sie in Worten einen entsprechenden Algorithmus, und analysieren Sie dessen asymptotische Laufzeit.

Hinweis: Ihre Lösung soll effizienter sein, als einfach den Algorithmus aus b) einmal für jeden möglichen Startort auszuführen!

Aufgabe 4:

Gegeben sei eine Kollektion von insgesamt n Zahlen, wobei die gleiche Zahl wiederholt vorkommen kann. Eine Kollektion ist also keine Menge, sondern eine "Multimenge". In dieser Aufgabe soll festgestellt werden, wie oft eine gegebene Zahl in der Kollektion vorkommt.
Beispiel: in der Kollektion 3, 4, 3, 5, 11, 0, 4, 12 bestehend aus $n = 8$ Zahlen kommen die Zahlen 3 und 4 je zweimal vor, die Zahlen 0, 5, 11 und 12 kommen je einmal vor, und alle anderen Zahlen kommen gar nicht vor (0-mal).

2 P a) Entwerfen Sie einen möglichst effizienten Algorithmus, der für eine gegebene Kollektion von n Zahlen alle Zahlen ausgibt, welche mindestens dreimal vorkommen. Welche asymptotische Laufzeit hat Ihre Lösung im schlechtesten Fall?

4 P b) Anstatt einmal alle dreifach vorkommenden Zahlen auszugeben, möchten wir nun eine Kollektion von Zahlen dynamisch verwalten. Wir suchen daher eine Datenstruktur, welche die folgenden drei Operationen unterstützt:

- `MindDreimal(`x`)`: gibt `TRUE` zurück, falls die Zahl x mindestens dreimal in der Kollektion vorkommt, und sonst `FALSE`.
- `Einfügen(`x`)`: fügt die Zahl x in die Kollektion ein (unabhängig davon, ob die Zahl schon vorkommt oder nicht).
- `Entfernen(`x`)`: löscht genau ein Vorkommen der Zahl x aus der Kollektion, falls x überhaupt in der Kollektion vorkommt.

Beschreiben Sie eine Datenstruktur, sowie die zugehörigen Algorithmen, welche diese drei Operationen effizient unterstützt. Geben Sie den Platzbedarf Ihrer Datenstrukutur und zudem für jede Operation die asymptotische Laufzeit an.

3 P c) Nun möchten wir eine andere Variante von Aufgabe b) lösen, bei der anstatt der Operation `MindDreimal(`x`)` die folgende Operation unterstützt wird: Für ein beliebiges k soll die Anfrage `Mal(`k`)` alle Zahlen ausgeben, welche mindestens k-mal in der Kollektion vorkommen. Beschreiben Sie eine Datenstruktur, sowie den zugehörigen Algorithmus, mit dem `Mal(`k`)` effizient beantwortet werden kann. Die Laufzeit von `Mal(`k`)` soll dabei output-sensitiv sein, das heisst, falls `Mal(`k`)` nur wenig Zahlen ausgeben muss, dann soll die Laufzeit kleiner sein, als wenn `Mal(`k`)` viele Zahlen ausgeben muss. Beschreiben Sie auch hier die asymptotische Laufzeit von `Mal(`k`)`, `Einfügen(`x`)` und `Entfernen(`x`)`, und geben Sie den Platzbedarf Ihrer Datenstruktur an.

Aufgabe 5:

Wie Sie wissen, gibt es in der Schweiz folgende Münzwerte: 5 Rappen, 10 Rappen, 20 Rappen, 50 Rappen, 1 Franken, 2 Franken, 5 Franken. Damit lassen sich alle Geldbeträge darstellen, deren Wert in Rappen durch 5 teilbar ist. In dieser Aufgabe befassen wir uns mit der Anzahl verschiedener Möglichkeiten, einen gegebenen Betrag durch irgendeine Menge von Münzen zu erreichen. Beispielsweise kann der Betrag von 20 Rappen auf vier verschiedene Arten erreicht werden:

(1) $5 + 5 + 5 + 5$ Rappen,

(2) $5 + 5 + 10$ Rappen,

(3) $10 + 10$ Rappen,

(4) 20 Rappen.

Beachten Sie, dass es dabei auf die Reihenfolge der Münzen nicht ankommt; 5+5+10 Rappen ist also dieselbe Menge von Münzen wie 5+10+5 Rappen. Sie sollen jetzt ein Programm entwickeln, mit dem man diese Anzahl Möglichkeiten für jeden beliebigen Geldbetrag und jede beliebige Kollektion von Münzwerten bestimmen kann. Die Eingabe besteht einerseits aus einem Array `M: ARRAY[INTEGER]` von Ganzzahlen, in dem alle Münzwerte in aufsteigender Reihenfolge gespeichert sind, und zwar natürlich in derselben Einheit (z.B. Rappen), und andererseits dem gewünschten Geldbetrag `B: INTEGER`. Für die Schweizer Währung hätte das Array also folgenden Inhalt:
`M := << 5, 10, 20, 50, 100, 200, 500 >>`

3 P a) Erstellen Sie ein rekursives Programm in Pseudocode, welches die Anzahl Möglichkeiten ausgibt, den Betrag `B` aus Münzwerten in `M` darzustellen. Geben Sie die Laufzeit Ihres Algorithmus an.

4 P b) Entwerfen Sie einen Algorithmus nach dem Muster der dynamischen Programmierung, der die Anzahl Möglichkeiten berechnet, den Betrag `B` aus Münzwerten in `M` darzustellen. Geben Sie die Laufzeit Ihres Algorithmus an.

1 P c) Beschreiben Sie in Worten, wie durch Rückverfolgung in der Lösungstabelle gemäss b) *eine* mögliche Darstellung des Betrags `B` gefunden werden kann, sofern überhaupt eine existiert. Geben Sie die Laufzeit für die Rückverfolgung an.