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
Jörg Derungs

Basisprüfung D-INFK

Datenstrukturen & Algorithmen

30. September 2004

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.

Viel Erfolg!

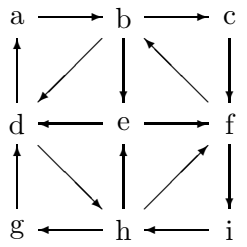
Stud.-Nummer: _____

Aufgabe	1	2	3	4	5	**Σ**
Mögl. Punkte	9	7	9	13	12	50
Punkte						

AUFGABE 1:

In dieser Aufgabe sollen Sie **nur** die Ergebnisse angeben. Diese können Sie direkt bei den Aufgaben notieren. 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.
Als Ordnung verwenden wir für Buchstaben die alphabetische Reihenfolge, für Zahlen die aufsteigende Anordnung gemäss ihrer Grösse.

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



1 P b) In der folgenden Tabelle ist ein Heap in der üblichen impliziten Form gespeichert:

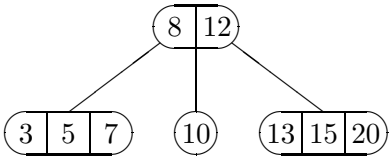
[3, 5, 7, 15, 10, 8, 20, 16, 20, 20, 13, 12]

Wie sieht die Tabelle aus, nachdem das kleinste Element gelöscht und die Heap-Bedingung wieder hergestellt wurde?

1 P c) Zeichnen Sie den binären Suchbaum, dessen Preorder-Traversierung die Folge
7, 6, 2, 1, 4, 3, 5, 12, 9, 8, 10, 11
ergibt.

1 P

d) Fügen Sie die Schlüssel 6 und 14 nacheinander in den untenstehenden B-Baum der Ordnung 4 ein.



1 P

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

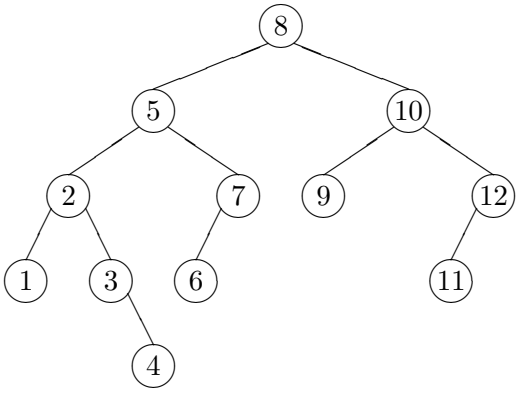
Schlüssel	a	b	c	d	e	f	g	h
Anzahl Zugriffe	4	6	11	7	8	3	10	2

1 P

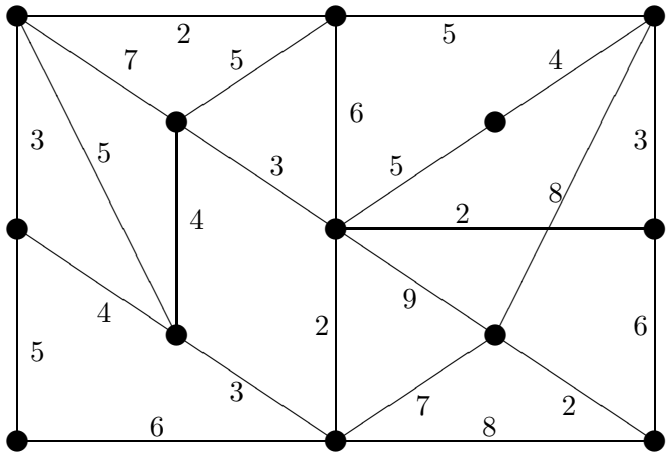
f) Fügen Sie die Schlüssel $46, 21, 37$ mittels Double Hashing in die Hashtabelle ein.
Die zu verwendenden Hash-Funktionen sind $h(k) = k \mod 13$ und $h'(k) = (k \mod 11) + 1$.

				30	18		20		9	75	24	12
0	1	2	3	4	5	6	7	8	9	10	11	12

1 P g) Löschen Sie den Schlüssel 7 aus dem AVL-Baum und balancieren Sie ihn wieder.



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



1 P i) Geben Sie an, wie die nach der Transpose-Regel selbstanordnende Liste aussieht, nachdem auf die Elemente 'N', 'E', 'T', 'T' in dieser Reihenfolge zugegriffen wurde.

D → A → T → E → N

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.

- $n!$
- $n^2 \, (\log n)^2$
- $\left(\frac{101}{99}\right)^n$
- $\frac{n^3+n+7}{n-3}$
- $123 \cdot n^{2.5}$
- $\frac{n^3}{\log n}$
- $n^{\log n}$
- $\frac{\sqrt{n}}{n}$

3 P b) Gegeben ist die folgende Rekursionsgleichung:

$$T(n) := \begin{cases} 9T(\frac{n}{3}) + 6n & n > 1 \\ 3 & n = 1 \end{cases}$$

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

Hinweise:

i) Sie können annehmen, dass n eine Potenz von 3 ist.

ii) $\sum_{i=0}^{k} q^i = \frac{q^{k+1}-1}{q-1}$.

1 P

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

```
from i := 1 until i > n loop
   from j := i until j > n loop
      c := c + j
      j := j * 2
   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 i := 1 until i > n loop
   from j := i until j > n loop
      s := 0
      from k := i until k := j loop
         s := s + a.item (k)
         k := k + 1
      end
      if s > max then max := s end
      j := j + 1
   end
   i := i + 1
end
```

Hinweis: `a.item` hat konstante Laufzeit.

1 P

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

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

AUFGABE 3:

Ein ideales Tanzpaar ist ein Paar, bei dem der Mann exakt fünf Prozent grösser ist als die Frau. Die Grössen der Mitglieder des Tanzklubs sind in zwei Arrays der Grösse nach gespeichert, die grössten mit dem kleinsten Index.

```
class TANZKLUB
feature -- Access
   damen: ARRAY [DOUBLE]
         -- Damen nach Groesse sortiert, groesste mit kleinstem Index
   herren: ARRAY [DOUBLE]
         -- Herren nach Groesse sortiert, groesster mit kleinstem Index

   paare: INTEGER
         -- von Ihnen zu implementierende Funktion
end -- class TANZKLUB
```

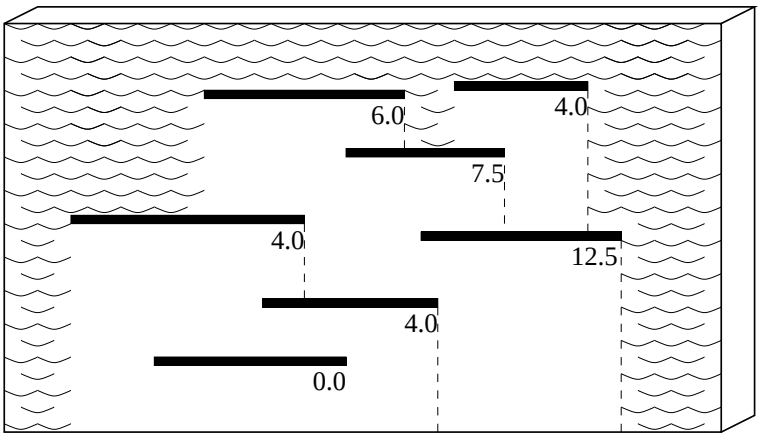
2 P a) Beschreiben Sie kurz in Worten einen Ansatz für einen Algorithmus, der möglichst effizient berechnet, wieviele ideale Tanzpaare gebildet werden können.

6 P b) Schreiben Sie in Eiffel eine möglichst effiziente Funktion `paare`, welche berechnet, wieviele ideale Tanzpaare gebildet werden können.

1 P c) Geben Sie die Laufzeit Ihrer Funktion in Abhängigkeit der Anzahl Damen (`N = damen.count`) und der Anzahl Herren (`M = herren.count`) an.

AUFGABE 4:

Ein Künstler baut eine Wasser-Installation. Über den oberen Rand einer Wand fliesst gleichmässig viel Wasser. An der Wand sind waagrechte Regenrinnen angebracht, welche das Wasser auffangen und nur an ihrem rechten Ende wieder abfliessen lassen.



Sie sollen nun mit einem Programm berechnen, wieviel Wasser pro Sekunde durch jede Regenrinne läuft.
Hinweise: Die Lage einer Regenrinne ist in einem REGENRINNE-Objekt gespeichert. Die Regenrinnen sind im Feature `rinnen: ARRAY [REGENRINNE]` in beliebiger Reihenfolge abgelegt. Die Anzahl der Regenrinnen (`rinnen.count`) ist `N`. Die Wassermenge, die pro Sekunde und Centimeter (in der Breite) über die Mauer fliesst, ist `amount: DOUBLE`.

```
class REGENRINNE
feature -- Access
   links: INTEGER
         -- Abstand des linken Endes der Regenrinne vom linken Rand
         -- der Wand, in cm
   rechts: INTEGER
         -- Abstand des rechten Endes der Regenrinne vom linken Rand
         -- der Wand, in cm
   hoehe: INTEGER
         -- Abstand der Regenrinne vom Boden, in cm
   wassermenge: DOUBLE
         -- von Ihnen zu setzender Wert

feature -- Element change
   set_wassermenge (w: DOUBLE)
         -- Verwenden Sie diese Prozedur, um die Loesung
         -- in 'wassermenge' zu speichern
      ensure
         wassermenge = w
end -- class CONTAINER
```

Für diese Aufgabe dürfen Sie die in der Vorlesung behandelten Datenstrukturen sowie die Prozedur `sort (a: ARRAY[ANY])` verwenden, ohne sie weiter zu definieren.

bitte wenden

3 P a) Beschreiben Sie in zwei bis drei kurzen Sätzen einen Ansatz für einen Algorithmus, der für jede Regenrinne berechnet, wieviel Wasser sie pro Sekunde durchläuft.

7 P b) Formulieren Sie in Pseudocode einen möglichst effizienten Algorithmus, der für jede Regenrinne berechnet, wieviel Wasser sie pro Sekunde durchläuft.

Geben Sie die Laufzeit Ihres Algorithmus in Abhängigkeit der Anzahl Regenrinnen an und begründen sie diese.

3 P c) Beschreiben Sie in Worten, wie Ihr Algorithmus erweitert werden muss, wenn das Wasser nicht am rechten Ende, sondern an einer in der Klasse `REGENRINNE` angegebenen Position `loch: INTEGER` abläuft.

Geben Sie die Laufzeit des erweiterten Algorithmus an und begründen Sie Ihr Resultat.

AUFGABE 5:

In der Hauptstadt wird eine Prachtstrasse der Länge `L` vom Parlament bis zum Hafen gebaut. Die Strasse soll in der Mitte einen marmornen Gehweg erhalten. Die zur Verfügung stehenden Marmorplatten haben alle die richtige Breite, aber unterschiedliche Längen. Sie sollen nun so ausgewählt werden, dass möglichst wenig Ausschuss entsteht, das heisst, die Summe der Längen der Marmorsteine soll mindestens `L` betragen, aber möglichst klein sein.

Die Länge der Strasse ist `L: INTEGER`.
Die Längen der Marmorplatten sind in einem Array `platten: ARRAY [INTEGER]` gegeben. Die Anzahl der Platten (`platten.count`) ist `N`.

2 P a) Erstellen Sie ein rekursives Programm in Pseudocode, welches den minimalen Ausschuss, also die minimale Differenz zwischen der Länge der für den Gehweg benutzten Platten und der Länge `L` der Strasse, berechnet.

Geben Sie die Laufzeit des rekursiven Algorithmus an.

5 P b) Entwerfen Sie einen Algorithmus nach dem Muster der dynamischen Programmierung, der den minimalen Ausschuss berechnet.

Geben Sie die Laufzeit Ihres Algorithmus an.

2 P c) Beschreiben Sie in Worten, wie durch Rückverfolgung die richtigen Marmorplatten für den Gehweg gefunden werden können.

Geben Sie die Laufzeit für die Rückverfolgung an.

3 P d) Entwerfen Sie einen Algorithmus nach dem Muster der dynamischen Programmierung, der den minimalen Ausschuss berechnet für den Fall, dass nicht ein Gehweg in der Mitte gebaut werden soll, sondern an jeder Strassenseite einer, also zwei je `L` lange Gehwege. Dabei darf der Ausschuss von der einen Seite nicht auf der anderen Seite benutzt werden.

Geben Sie die Laufzeit Ihres Algorithmus an.