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

Datenstrukturen & Algorithmen

5. September 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 irgendjemanden alles löst!

Viel Erfolg!

Stud.-Nummer: _____

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

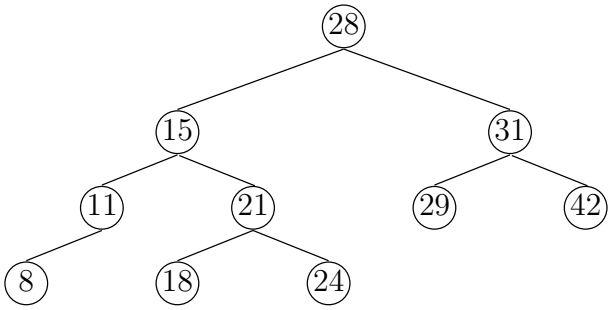
AUFGABE 1:

Hinweise:

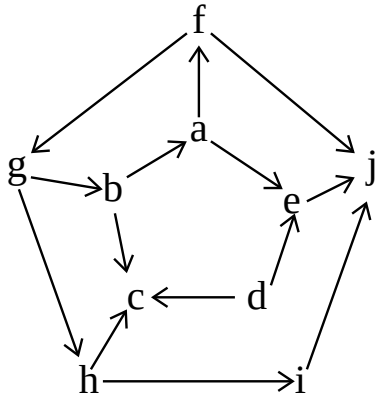
1. In dieser Aufgabe sollen Sie **nur** die Ergebnisse angeben. Diese können Sie direkt bei den Aufgaben notieren.
2. Sofern Sie die Notationen, Algorithmen und Datenstrukturen aus der Vorlesung "Datenstrukturen & Algorithmen" verwenden, sind Erklärungen oder Begründungen nicht notwendig. Falls Sie jedoch 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.
$[1, 3, 5, 12, 6, 8, 11, 14, 13, 17, 42, 16, 15]$
Wie sieht die Tabelle aus, nachdem das kleinste Element entfernt wurde und die Heap-Bedingung wieder hergestellt wurde?

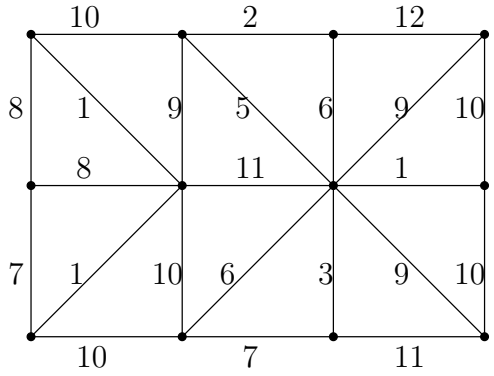
1 P b) Fügen Sie den Schlüssel 16 in den AVL-Baum ein und balancieren Sie ihn wieder falls nötig.



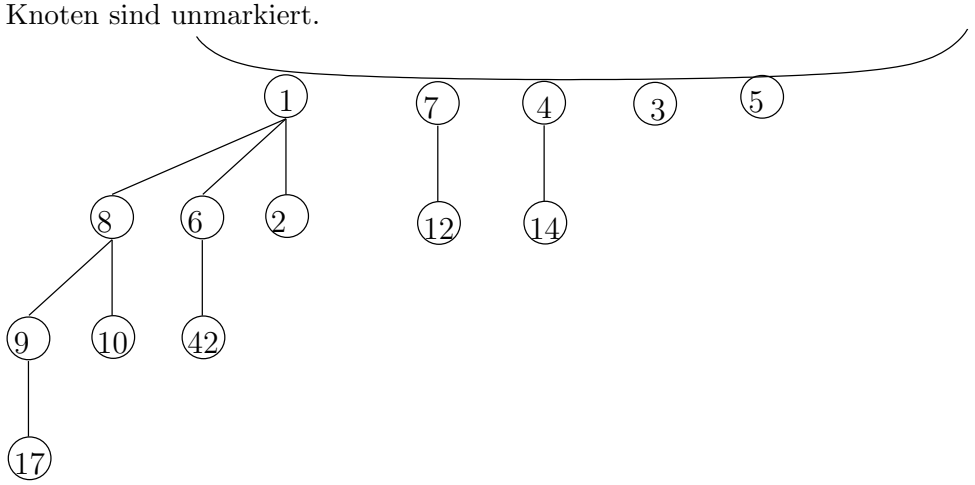
1 P c) Der folgende gerichtete Graph wird mit Breitensuche traversiert. Die Suche startet beim Knoten a. Geben Sie eine Reihenfolge an, in der die Knoten erreicht werden können.



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



1 P e) Führen Sie auf dem unten stehenden Fibonacci-Heap die Operation Delete-Min aus. Alle Knoten sind unmarkiert.

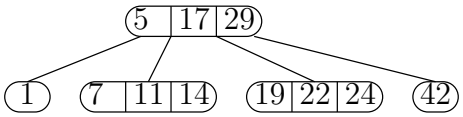


1 P f) Fügen Sie die Schlüssel $15, 19, 14, 7$ in dieser Reihenfolge mittels Double Hashing in die Hashtabelle ein.

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

11	45	24	47	04	38					
0	1	2	3	4	5	6	7	8	9	10

1 P g) Löschen Sie zuerst den Schlüssel 42 aus dem unten stehenden B-Baum der Ordnung 4, und fügen Sie danach den Schlüssel 12 ein, jeweils mit den zugehörigen Strukturänderungen.



Nach Löschen von 42:	Nach Einfügen von 12:

1 P h) Geben Sie an, wie die nach der Move-To-Front-Regel selbstanordnende Liste aussieht, nachdem auf die Elemente 'A', 'L', 'G', 'O' in dieser Reihenfolge zugegriffen wurde.

$$H \rightarrow O \rightarrow L \rightarrow G \rightarrow A$$

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	6	15	22	9	25	9	4	10

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.

- 2^{n^2}
- $\frac{n\sqrt{n}}{5}$
- n^n
- $\sum_{i=1}^{n} i$
- $12n$
- $n \log n$
- $n!$

3 P b) Gegeben ist die folgende Rekursionsgleichung:

$$T(n) := \begin{cases} 9T(\frac{n}{3}) + 6n + 8 & n > 1 \\ 5 & n = 1 \end{cases}$$

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 3 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:= 1 until i > (n // 4) loop
   from j:= 2*i until j > n 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 i:= n until i < 1 loop
  from j:= i until j > n loop
     j := j * 2
  end
  i:= i - 1
end
```

1 P

e) 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
   from j:=1 until j > i loop
      k := k + 1
      j:= j + 1
   end
   i:= 2*i
end
```

AUFGABE 3:

Gegeben sei eine Menge M von paarweise verschiedenen positiven ganzen Zahlen, die unsortiert in einem Array der Grösse n gespeichert sind. Wir möchten alle verschiedenen Tripel von Zahlen der Menge M finden, bei denen die Summe von zwei Zahlen gleich der dritten Zahl ist. Genauer: wir möchten alle Tripel (a, b, c) finden, für die $a + b = c, \{a, b, c\} \subset M$ gilt, wobei wir $a < b$ verlangen, damit jedes Tripel nur einmal vorkommen kann.
Für die Zahlenmenge im Array $A = [8, 2, 5, 3, 13, 6]$ gibt es vier solche Tripel: $(2, 3, 5), (2, 6, 8), (3, 5, 8)$ und $(5, 8, 13)$.

Die Zahlen sind in einem Array:

```
numbers: ARRAY [ INTEGER ]
```

gespeichert, mit `numbers.count`$= n$.

3 P a) Beschreiben Sie kurz in Worten einen Ansatz für einen Algorithmus, der aus einem gegebenen Array `numbers` von ganzen Zahlen möglichst effizient alle Tripel mit der oben gegebenen Eigenschaft findet.

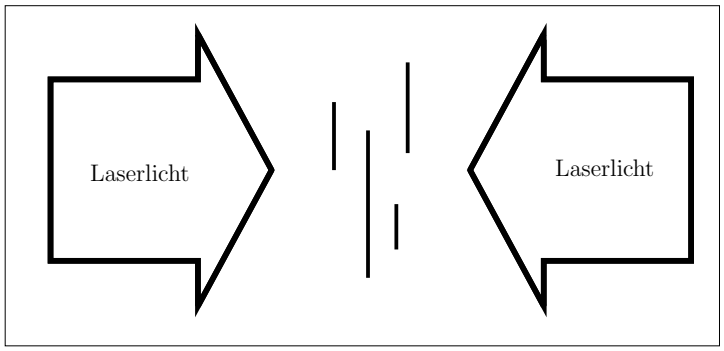
5 P b) Schreiben Sie in Eiffel eine möglichst effiziente Funktion, welche die verlangten Tripel nach dem in Teilaufgabe a) beschriebenen Algorithmus bestimmt, und das Resultat in einem geeigneten Array zurückgibt.

1 P c) Geben Sie die Laufzeit Ihrer Funktion in Abhängigkeit von $n =$ `numbers.count` an.

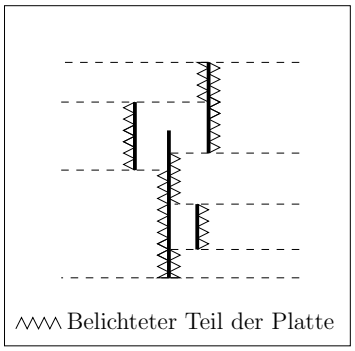
Hinweise zu Aufgabeteil b): Sie dürfen die in Eiffel vorhandenen Datenstrukturen `STACK[Q]`, `QUEUE[K]`, sowie einen effizienten Algorithmus `sort( a: ARRAY [ COMPARABLE ] )` zum Sortieren (in aufsteigender Reihenfolge) und eine effiziente Funktion zum Suchen `find(key: COMPARABLE, a : ARRAY[ COMPRABLE ])` in einem aufsteigend sortierten Array benutzen, ohne diese weiter zu implementieren oder zu definieren. Dabei gibt `find(key: COMPARABLE, a : ARRAY[ COMPRABLE ])` den Array-Index von `key` im Array `a` zurück, falls die Suche erfolgreich terminiert, und `a.lower -1` falls `key` nicht in `a` vorkommt.

AUFGABE 4:

Wir betrachten ein Kunstwerk, bei dem vertikale Platten mit Laserlicht von der Seite bestrahlt werden. Weil das Licht die Platten erhitzt, muss jede Platte gekühlt werden, und zwar umso mehr, je mehr Licht auf diese trifft. Das eintreffende Licht heizt die Platten von beiden Seiten auf.



Input



Bestrahlung

Die Platten haben verschiedene Positionen und Höhen, sind aber alle gleich tief (in der dritten Dimension, in der Skizze nicht zu sehen). Falls eine Platte auf einer bestimmten Höhe vom Laserlicht belichtet wird, so ist die Platte auf dieser Höhe über die gesamte Tiefe belichtet. Deshalb können wir das Problem als ein zweidimensionales Problem betrachten.
Die Platten sind vertikal ausgerichtet. Links und rechts der Platten befinden sich flächenförmige Laserlichtquellen, welche über die gesamte Höhe des Kunstwerks horizontale Lichtstrahlen aussenden. Licht, welches auf eine Platte trifft, wird vollständig absorbiert.
Die Koordinaten der Platten sind gegeben durch den Abstand vom linken Rand, den Abstand vom Boden, und schliesslich die Höhe der Platte. Alle Koordinaten sind paarweise verschieden.

```
class PLATTE
feature -- ACCESS
   abstand: INTEGER
    -- Abstand der Platte vom linken Rand in cm
   unten: INTEGER
    -- Abstand des unteren Endes der Platte vom Boden in cm
   hoehe: INTEGER
    -- Hoehe der Platte in cm
```

Die n Platten sind in einem Array `Werk: ARRAY [ PLATTE ]` gespeichert.

3 P a) Beschreiben Sie kurz in Worten einen Algorithmus, der möglichst effizient für jede Platte die gesamte bestrahlte Höhe berechnet (als Summe der von links bestrahlten Höhe und der von rechts bestrahlten Höhe).

1 P b) Geben Sie die Laufzeit Ihres Algorithmus aus a) in Abhängigkeit von der Anzahl Platten n an.

3 P c) Beschreiben Sie Ihre Lösung für Teilaufgabe a) in einem Eiffel-ähnlichen Pseudocode.

3 P d) Da sich die Platten zu stark erhitzen, wird das Laserlicht nicht mehr über die gesamte Höhe horizontal ausgestrahlt. Stattdessen werden die Platten nur mit mehreren einzelnen horizontalen Strahlen (der Höhe 0) von links und rechts und auf unterschiedlichen Höhen bestrahlt. Beschreiben Sie in Worten einen möglichst effizienten Algorithmus, der für jede Platte die Anzahl eintreffender Laser-Strahlen berechnet. Geben Sie die Laufzeit Ihres Verfahren an.

AUFGABE 5:

In einer Mond-Mission wird ein Roboter eingesetzt. Dieser ist fehlgelandet und befindet sich nun in gefährlichem Terrain. Die Mission-Control-Ingenieure sollen nun den Roboter unbeschädigt zu seinem geplanten Einsatzort leiten.
Der Roboter bewegt sich auf einem zweidimensionalen Gitter mit m Zeilen und n Spalten. Jede Zelle des Gitters hat ein Risiko, angegeben als eine positive ganze Zahl. Je höher das Risiko, desto grösser ist die Wahrscheinlichkeit, dass der Roboter in dieser Zelle einen Schaden erleidet, der die Weiterfahrt unmöglich macht. Die Ingenieure interessieren sich für die Summe der Risiken der Zellen eines Weges. Sie nennen diese Summe das *Gesamtrisiko* des Weges.
Der Roboter bewegt sich in jedem Schritt entweder um eine Zelle nach rechts oder um eine Zelle nach unten. Der Weg muss von Zelle $(1, 1)$ starten (die Landeposition, oben links) und in Zelle (m, n) enden (der Einsatzort, unten rechts). Gesucht ist ein Weg für den Roboter durch die Zellen des Gitters, der das Gesamtrisiko minimiert. Die Ingenieure nennen einen Weg mit diesen Eigenschaften einen *sichersten* Weg. Ein Beispiel ist in der Abbildung links unten angegeben.

	1						$n = 7$
1		5	2	12	8	4	1
	1	3	8	10	4	5	1
	1	1	2	8	2	8	1
	7	5	2	4	4	6	1
$m = 5$	2	9	10	10	2	2	

Beispiel (schattiert) eines sichersten Weges für die gegebenen Risiken. Der Weg hat Gesamtrisiko 19.

Die Risiken sind in einem zweidimensionalen Array `risk: ARRAY2 [INTEGER]` gespeichert.

2 P a) Erstellen Sie ein rekursives Programm in Pseudocode, welches das Gesamtrisiko eines sichersten Weges berechnet.

Geben Sie die Laufzeit des rekursiven Algorithmus an.

5 P b) Entwerfen Sie einen Algorithmus nach dem Muster der dynamischen Programmierung, der das Gesamtrisiko eines sichersten Weges berechnet.

Geben Sie die Laufzeit Ihres Algorithmus an.

1 P c) Beschreiben Sie in Worten, wie durch Rückverfolgung in der Lösungstabelle gemäss b) die traversierten Zellen eines sichersten Weges gefunden werden können.

Geben Sie die Laufzeit für die Rückverfolgung an.

2 P d) Kann das Problem immer noch mittels dynamischer Programmierung gelöst werden, wenn sich der Roboter in jedem Schritt entweder um eine Zelle nach rechts, links, unten oder oben bewegen kann? Begründen Sie Ihre Antwort.