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

Zwischenklausur Datenstrukturen und Algorithmen, D-INFK

Datenstrukturen & Algorithmen

2. Juni 2005

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 90 Minuten.

Viel Erfolg!

Stud.-Nummer: _____

Aufgabe	1	2	3	4	$\mathbf{\Sigma}$
Mögl. Punkte	5	7	7	8	
Punkte					

AUFGABE 1:

Es sind sich viele Fans einig, dass unter den vorhandenen Eiskunstlauf-Disziplinen der Paarlauf die schönste ist, denn er kombiniert Harmonie, Kraft und Eleganz.
Natürlich ist es für das Auge am schönsten, wenn die Körpergrössen bei jedem Paar möglichst gut übereinstimmen. In dieser Aufgabe sollen Sie ein Eiffel-Programm schreiben, das für eine Gruppe von Läuferinnen und Läufern das schönste Paar bildet. Dabei soll der Grössenunterschied, der bei dem Paar auftritt, minimal sein über alle mögliche Paare.
Die Angaben einer Person sind in der Klasse `PERSON` gespeichert:

```
class PERSON
feature -- Access
   height: INTEGER
        -- die Groesse der Person in Zentimeter
   name: STRING
        -- der Name der Person
```

Die Paare werden jeweils aus einer Dame und einem Herrn gebildet. Die Damen sind im Array `damen: ARRAY [PERSON]`, die Herren im Array `herren: ARRAY [PERSON]` gespeichert, jeweils in aufsteigender Grösse.

2 P a) Beschreiben Sie kurz in Worten einen Ansatz für einen Algorithmus, der möglichst effizient berechnet, wie man aus den gegebenen Listen das schönste Paar bildet.

2 P b) Schreiben Sie in Eiffel eine möglichst effiziente Funktion `paar`, welche das schönste Paar bildet, und es auf der Konsole ausgibt.

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 2:

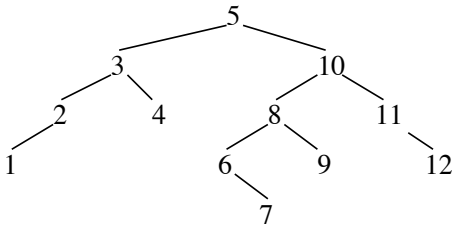
In dieser Aufgabe sollen Sie **nur** die Ergebnisse angeben. Diese können Sie direkt bei den Aufgaben notieren. Sofern Sie die Notationen, Algorithmen und Datenstrukturen aus der Vorlesung "Datenstrukturen & Algorithmen" verwenden, sind Erklärungen oder Begründungen nicht notwendig. Falls Sie jedoch 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) 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 b) Löschen Sie den Schlüssel 3 aus dem AVL-Baum und balancieren Sie ihn wieder.



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

A → L → G → O → R → I → T → H → M → U → S

1 P

d) Fügen Sie die Schlüssel $12, 3, 26, 16, 22$ mittels Double Hashing in die Hashtabelle ein. Die zu verwendenden Hash-Funktionen sind $h(k) := k \mod 7$ und $h'(k) := (k \mod 5) + 1$.

0	1	2	3	4	5	6

1 P

e) Beim Einfügen eines Elements in eine Datenstruktur unterscheiden wir zwischen Vergleichen, die benötigt werden, um die richtige Stelle für das neue Element zu finden, und Schreiboperationen, die benötigt werden, um die Datenstruktur so abzuändern, dass sie das neue Element enthält.

Geben Sie die Anzahl Vergleiche und die Anzahl Schreiboperationen in Theta-Notation für das möglichst effiziente Einfügen eines Elements in folgende Datenstrukturen (welche bereits `n` Elemente enthalten) an:

	Vergleiche	Schreib-Operationen
sortierter Array		
sortierte lineare Liste		
AVL-Baum		
Heap		

1 P

f) Unter einem ternären Baum versteht man einen Baum, bei dem jeder Knoten, der kein Blatt ist, genau drei Nachfolger hat. Wieviele Knoten hat ein vollständiger ternärer Baum auf dem h-ten Niveau? Die Wurzel steht per Definition auf Niveau 0.

1 P

g) Zeichnen Sie den binären Suchbaum, dessen Knoten in preorder-Reihenfolge traversiert diese Zahlenfolge ergibt:

$5, 4, 2, 1, 3, 9, 6, 8, 7, 11, 10, 12$

AUFGABE 3:

3 P a) Gegeben ist die folgende Rekursionsgleichung:

$$T(n) := \begin{cases} 4T(\frac{n}{2}) - 5 & n > 1 \\ 3 & n = 1 \end{cases}$$

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

1 P b) 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.

- $7n^2 + 3n$
- $\frac{n^3}{7} - 7n^2$
- 2^n
- $\frac{n^3}{\log n}$
- $n!$
- $n^2 \log n$
- $2^{\sqrt{n}}$

1 P c) Markieren Sie in der folgenden Liste genau die Aussagen, die korrekt sind:

- ☐ $n^5 - n \in O(n^4)$
- ☐ $\log_3 n + \log_2 n \in O(\log_2 n)$
- ☐ $\sum_{i=1}^{n}(ni) \in \Omega(n^3)$
- ☐ $(n+1)! \in \Theta(n!)$

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 := 1 until j > i loop
      j := j + 1
   end
   i := i // 2
end
```

1 P

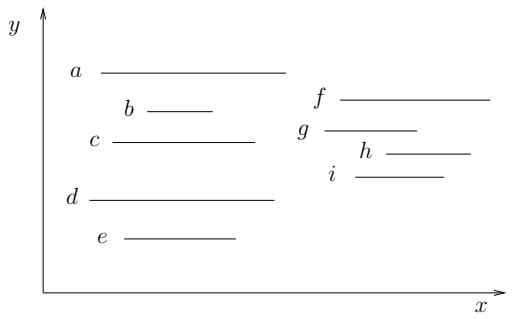
e) Geben Sie die asymptotische Laufzeit in Abhängigkeit von n für folgenden Algorithmus in Theta-Notation an:

```
from i := 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.

AUFGABE 4:

Gegeben sei eine Menge von n horizontalen Segmenten in der Ebene. Wir nennen ein Paar von horizontalen Segmenten a und b *sichtbar*, wenn man a mit einem (gedachten) vertikalen Liniensegment mit b verbinden kann, ohne dass dieses andere zwischen a und b vorhandene Segmente schneidet. Für das Beispiel in der Abbildung sind die horizontalen Segmentpaare $(a,b),(a,c),(a,d),(b,c),(c,d)$, $(d,e),(f,g),(f,h),(g,h),(g,i),(h,i)$ sichtbar. Die Pfeile x, y beschreiben das Koordinatensystem, bei dem der Kreuzungspunkt die Koordinaten $(0,0)$ hat.



Ein Segment ist wie folgt definiert:

```
class SEGMENT
feature -- Access
   x: INTEGER
        -- x Koordinate des linken Endpunkts des Segments.
   y: INTEGER
        -- y Koordinate des linken Endpunkts des Segments.
   l: INTEGER
        -- Laenge des Segments
```

Der linke Endpunkt eines Segments hat also die Koordinaten (x,y), der rechte Endpunkt die Koordinaten $(x+l,y)$. Die Menge der Segmente ist im Array `segments: ARRAY [SEGMENT]` gespeichert.
Hinweise:

i) Sie können annehmen, dass sich keine zwei Segmente berühren oder überlappen. Weiter sind die x-Koordinaten aller Anfangs- und Endpunkte der horizontalen Segmente (paarweise) verschieden.

ii) Für diese Aufgabe dürfen Sie die in der Vorlesung behandelten Datenstrukturen sowie die Algorithmen zum Sortieren verwenden, ohne sie weiter zu definieren.

3 P a) Beschreiben Sie in wenigen Sätzen, wie man die Paare sichtbarer Segmente möglichst effizient finden kann.

3 P b) Schreiben Sie eine Prozedur in Pseudocode, welche möglichst effizient die Paare sichtbarer Segmente aus dem Array `segments` berechnet.

1 P c) Geben Sie die Laufzeit und den Speicherbedarf Ihres Verfahrens an.

1 P d) Beschreiben Sie, wie Sie Ihren Lösungsansatz adaptieren müssen, wenn man zusätzlich zum Sichtbarkeitsproblem von horizontalen Segmenten auch das Sichtbarkeitsproblem von vertikalen Segmenten lösen will. Die Sichtbarkeit für vertikale Segmente ist, analog zur Sichtbarkeit von horizontalen Segmenten, über horizontale verbindende Liniensegmente definiert. Für die vertikalen Segmente gelten dieselben Einschränkungen bezüglich der Position wie bei horizontalen Segmenten.