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
Holger Flier

Prüfung
Datenstrukturen und Algorithmen
D-INFK

26. August 2010

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	9	9	9	43
$\sum$ 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) Führen Sie auf dem folgenden Array den ersten Durchgang des Sortierverfahrens *Natürliches 2-Wege-Mergesort* durch.

5	10	18	14	12	0	3	15	9	4	6

nach dem ersten Durchgang:

1 P b) Geben Sie die kürzeste Folge von Zugriffen an, die, ausgehend von der Liste

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

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

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

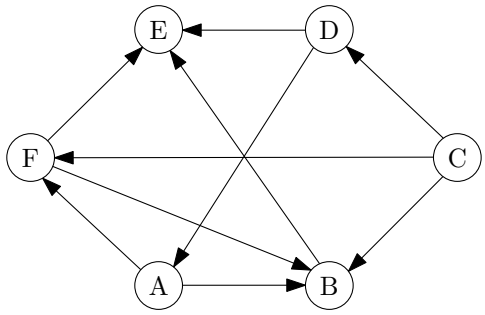
1 P c) Fügen Sie die Schlüssel $11, 15, 23, 21, 8, 16$ in dieser Reihenfolge mittels Offenem Hashing in die folgende Hashtabelle ein, und benutzen Sie dabei Double Hashing. Die zu verwendende Hash-Funktion ist $h(k) = k \mod 7$, und für das Sondieren soll die Hashfunktion $h'(k) = 1+(k \mod 5)$ benutzt werden.

0	1	2	3	4	5	6 (Index)

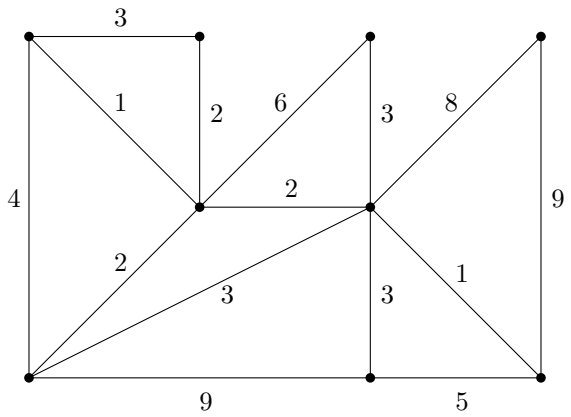
1 P d) Das untenstehende Array enthält die Elemente eines Min-Heaps in der üblichen Form gespeichert. Wie sieht das Array aus, nachdem das Minimum entfernt wurde und die Heap-Bedingung wieder hergestellt wurde?

2	6	9	14	11	12	16	15	18
1	2	3	4	5	6	7	8	9

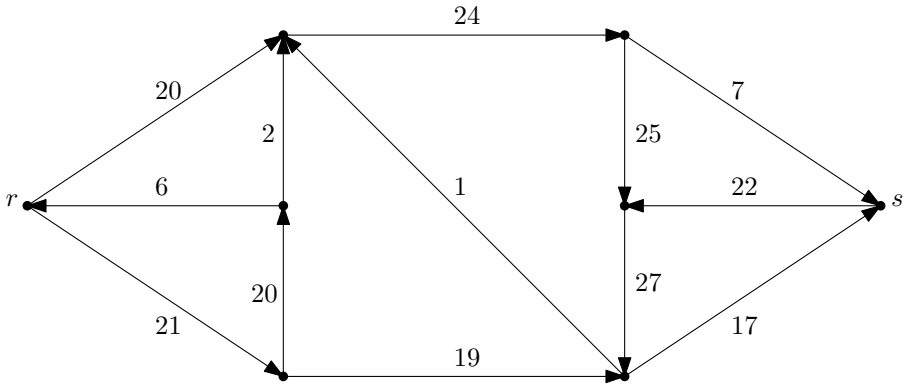
1 P e) Geben Sie für den folgenden azyklischen Graphen eine topologische Sortierung an.



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

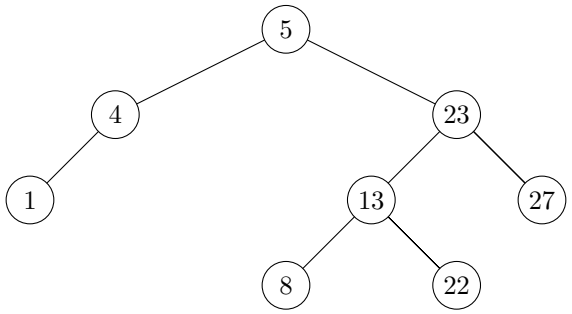


1 P g) Geben Sie den Wert eines maximalen (r, s)-Flusses in folgendem Netzwerk an und zeichnen Sie den entsprechenden minimalen Schnitt ein. Die Zahlen neben den Kanten geben die jeweilige Kantenkapazität an.



1 P

h) Fügen Sie in den untenstehenden AVL-Baum den Schlüssel 6 ein, und löschen Sie im entstandenen AVL-Baum den Schlüssel 23.



Nach Einfügen von 6:	Nach Löschen von 23:

1 P

i) Sei T der Splay-Tree, dessen Preorder-Traversierung die Werte $4, 3, 1, 2, 5, 7, 6, 8$ liefert. Zeichnen Sie den Splay-Tree, der entsteht, nachdem man den Knoten 7 aus T gelöscht hat (und die zugehörigen Splay-Operationen ausgeführt wurden).

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.

- $n^{\frac{3}{2}}$
- $\sqrt{n}$
- $n!$
- $\log(n)$
- $n^{\sqrt{n}}$
- $\log(n^n)$

3 P b) Gegeben ist die folgende Rekursionsgleichung:

$$T(n) := \begin{cases} 2T(\frac{n}{8}) + 1 & n > 1 \\ 1 & n = 1 \end{cases}$$

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

Hinweise:
(1) Sie können annehmen, dass n eine Potenz von 8 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 $n \in \mathbb{N}$ für folgenden Algorithmus in Theta-Notation an (Sie müssen Ihre Antwort nicht begründen):

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

1 P

d) Geben Sie die asymptotische Laufzeit in Abhängigkeit von $n \in \mathbb{N}$ für folgenden Algorithmus in Theta-Notation an (Sie müssen Ihre Antwort nicht begründen):

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

1 P

e) Geben Sie die asymptotische Laufzeit in Abhängigkeit von $n \in \mathbb{N}$ für folgenden Algorithmus in Theta-Notation an (Sie müssen Ihre Antwort nicht begründen):

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

Aufgabe 3:
Die Piraten des berüchtigten Schiffs Playfair haben auf einer entlegenen Insel einen Schatz gefunden, der aus einer Kiste mit n Münzen besteht. Jede Münze i, $1 \leq i \leq n$, hat einen ganzzahligen, positiven Wert w_i. Nun soll der Schatz gleichmässig unter allen Piraten aufgeteilt werden.

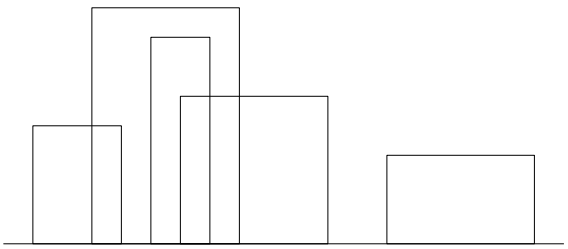
4 P a) Nehmen Sie zunächst an, dass der Schatz zwischen nur zwei Piraten aufgeteilt werden soll. Entwerfen Sie einen möglichst effizienten Algorithmus nach dem Muster der dynamischen Programmierung, der die Frage beantwortet, ob es eine gerechte Aufteilung der Münzen gibt. Genauer: Gibt es eine Menge $A \subset \{1, \ldots, n\}$ und eine Menge $B = \{1, \ldots, n\} \setminus A$ von Münzen, so dass $\sum_{i \in A} w_i = \sum_{i \in B} w_i$. Beschreiben Sie Ihren Algorithmus in Worten und/oder Pseudocode. Geben Sie die Laufzeit Ihres Algorithmus an.

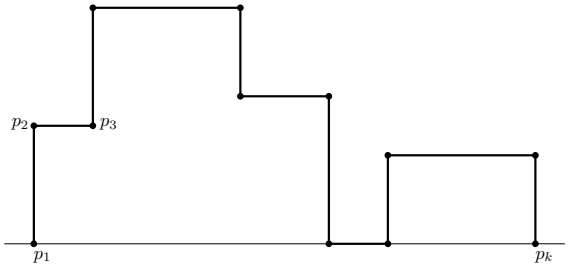
1 P b) Beschreiben Sie in Worten, wie aus der Lösungstabelle aus a) die Mengen A und B bestimmt werden können.

2 P c) Nehmen Sie nun an, es gäbe k Piraten, unter denen der Schatz gleichmässig aufgeteilt werden soll. Beschreiben Sie in Worten, wie der Algorithmus aus a) erweitert werden kann, um dieses Problem zu lösen. Geben Sie die Laufzeit Ihres Algorithmus an.

2 P d) Nehmen Sie nun an, dass $k-1$ Piraten von Kannibalen zum Abendessen "eingeladen" wurden. Ein Pirat bleibt übrig, und möchte mit dem Ruderboot gerne vor dem Abendessen abreisen. Das Ruderboot kann neben dem Piraten maximal ein Gewicht G laden. Der Pirat will nun aus den Münzen, die alle mit einer Gewichtsangabe g_i, $1 \leq i \leq n$, versehen sind, eine Menge S auswählen, so dass $\sum_{i \in S} g_i \leq G$ und $\sum_{i \in S} w_i$ maximiert wird. Weil dem Piraten aber nur sehr wenig Zeit bleibt, ist er auch mit einer Approximation zufrieden, die ihm eine Menge T von Münzen liefert, so dass $\sum_{i \in T} w_i \geq \frac{\sum_{i \in S} w_i}{2}$. Empfehlen Sie dem Piraten einen möglichst effizienten Algorithmus, und beschreiben Sie diesen Algorithmus kurz in Worten.

Aufgabe 4:
In dieser Aufgabe geht es um die Berechnung des Umrisses von Skylines und Bergketten.
Die Skyline einer Stadt (der Umriss) ergibt sich wie der Schatten einer Menge rechteckiger Hochhäuser, die allesamt auf dem Boden aufsitzen und als orthogonale Recktecke wie im Bild links gegeben sind. Der Umriss ist das orthogonale Polygon, das die Vereinigung aller solchen gegebenen Rechtecke beschreibt, wie im Bild rechts zu sehen. Der Umriss kann durch eine Menge von Punkten $p_1, p_2, \ldots, p_k$ beschrieben werden. Im Bild rechts sind einige dieser Punkte exemplarisch beschriftet.

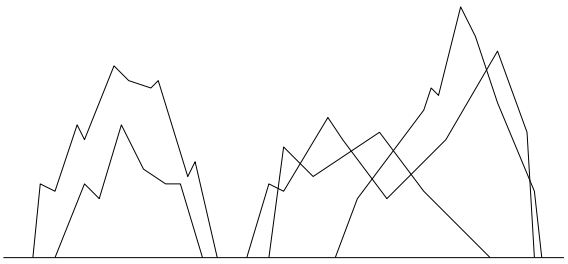


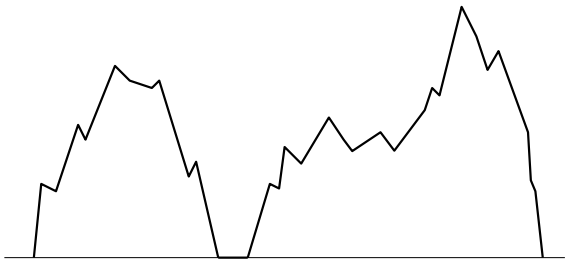


4 P a) Entwerfen Sie einen möglichst effizienten Algorithmus, der für eine wie oben beschriebene Menge von n Rechtecken den Umriss berechnet. Beschreiben Sie Ihren Algorithmus in Worten und/oder Pseudocode.

1 P b) Geben Sie die Laufzeit Ihres in a) entwickelten Algorithmus an.

Betrachten Sie nun folgende Verallgemeinerung des Problems, in der nicht mehr Rechtecke gegeben sind, sondern Bergketten, die als einfache Polygonzüge gegeben sind. Jeder einzelne Polygonzug ist monoton in x-Richtung, hat also keine überhängenden Stücke. Zwei Polygonzüge können sich ansonsten beliebig überschneiden oder ineinander enthalten sein. Ferner liegen alle Polygone mit der unteren Seite auf einer Linie, wie im Bild links zu sehen. Der gesuchte Umriss ist im rechten Bild dargestellt.





3 P c) Entwerfen Sie einen möglichst effizienten Algorithmus, der für eine wie oben beschriebene Menge von n Polygonen den Umriss berechnet. Beschreiben Sie Ihren Algorithmus in Worten und/oder Pseudocode.

1 P d) Geben Sie die Laufzeit Ihres in c) entwickelten Algorithmus an.

Aufgabe 5:

In dieser Aufgabe geht es um auf- und absteigende Teilfolgen von Zahlen. Gegeben ist eine Folge von n ganzen, positiven Zahlen $a_1, a_2, \ldots, a_n$, die paarweise verschieden sind, d.h. $a_i \neq a_j$ für $i \neq j$.

Eine *längste absteigende Teilfolge* von $a_1, a_2, \ldots, a_n$ ist eine Folge von absteigenden Zahlen, also eine Folge $a_{i(1)}, a_{i(2)}, \ldots, a_{i(k)}$, mit $1 \leq i(j) < i(j+1) \leq n$ für alle $1 \leq j < k$, so dass $a_{i(j)} > a_{i(j+1)}$ für alle $1 \leq j < k$ und k maximal ist.

Eine *längste alternierende Teilfolge* von $a_1, a_2, \ldots, a_n$ ist eine Folge von abwechselnd auf- und absteigenden Zahlen, also eine Folge $a_{i(1)}, a_{i(2)}, \ldots, a_{i(k)}$, mit $1 \leq i(j) < i(j+1) \leq n$ für alle $1 \leq j < k$, so dass k maximal ist und

- entweder $a_{i(j-1)} < a_{i(j)}$ für alle geraden j, $2 \leq j \leq k$, und $a_{i(j-1)} > a_{i(j)}$ für alle ungeraden j, $3 \leq j \leq k$, gilt (also $a_{i(1)} < a_{i(2)}$, $a_{i(2)} > a_{i(3)}$, $a_{i(3)} < a_{i(4)}$, usw.),
- oder es gilt $a_{i(j-1)} > a_{i(j)}$ für alle geraden j, $2 \leq j \leq k$, und $a_{i(j-1)} < a_{i(j)}$ für alle ungeraden j, $3 \leq j \leq k$ (also $a_{i(1)} > a_{i(2)}$, $a_{i(2)} < a_{i(3)}$, $a_{i(3)} > a_{i(4)}$, usw.).

Sei beispielsweise die Folge 8, 14, 16, 3, 4, 6, 7, 5, 18, 9, 1 gegeben. Dann ist 16, 7, 5, 1 eine längste absteigende Folge, und 14, 16, 3, 7, 5, 18, 1 eine längste alternierende Folge.

4 P a) Entwickeln Sie einen möglichst effizienten Algorithmus, der für eine wie oben beschriebene Folge von Zahlen eine längste absteigende Teilfolge berechnet. Beschreiben Sie Ihren Algorithmus in Pseudocode.

1 P b) Geben Sie die Laufzeit Ihres Algorithmus aus a) an.

3 P c) Entwickeln Sie einen möglichst effizienten Algorithmus, der für eine wie oben beschriebene Folge von Zahlen eine längste alternierende Teilfolge berechnet. Beschreiben Sie Ihren Algorithmus in Pseudocode.

1 P d) Geben Sie die Laufzeit Ihres Algorithmus aus c) an.