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, Beat Gfeller

Prüfung
Datenstrukturen und Algorithmen
D-INFK

10. Februar 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: _____

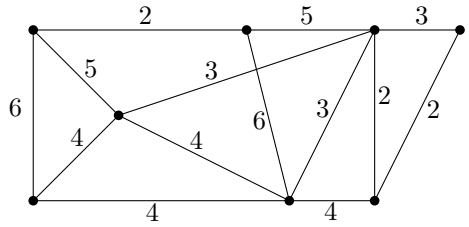
Aufgabe	1	2	3	4	5	**Σ**
Mögl. Punkte	9	7	9	9	9	43
Punkte						

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



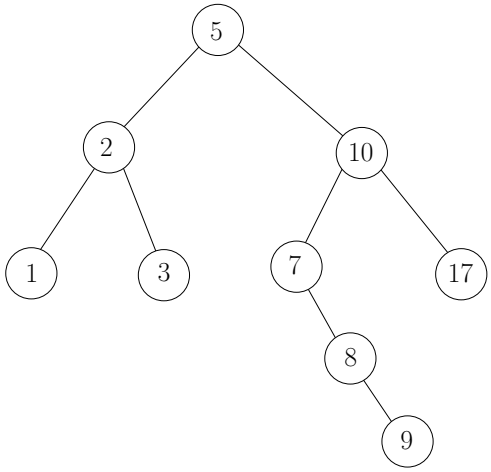
1 P b) Zeichnen Sie den binären Suchbaum, dessen Preorder-Traversierung die Folge $19, 9, 2, 7, 12, 17, 14, 13, 18, 21$ ergibt.

1 P c) Führen Sie auf dem folgenden Array den ersten Durchgang des Sortierverfahrens *Natürliches 2-Wege-Mergesort* durch.

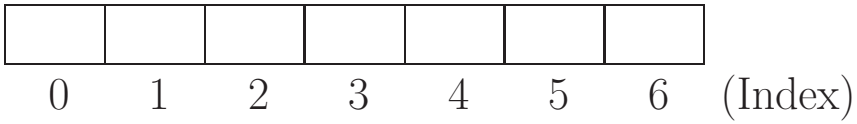
5	12	14	2	6	9	1	11	4	8	7

nach dem ersten Durchgang:

1 P d) Zeichnen Sie den Splay-Tree, der aus dem unten gezeigten entsteht, nachdem man auf den Knoten 9 zugegriffen hat (und die zugehörigen Splay-Operationen ausgeführt wurden).



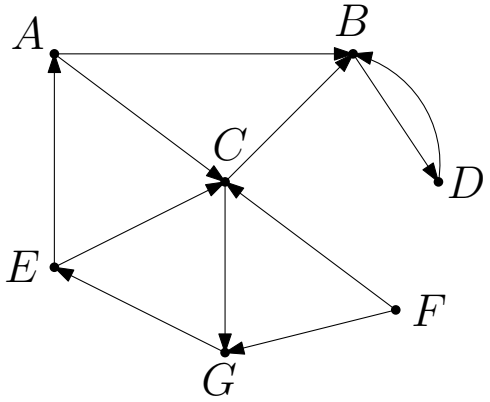
1 P e) Fügen Sie die Schlüssel $4, 8, 16, 14, 1, 9$ in dieser Reihenfolge mittels Offenem Hashing in die folgende Hashtabelle ein, und benutzen Sie dabei Double Hashing. Die zu verwendende Hash-Funktion ist $h(k) = k \mod 7$, und für das Sondieren soll die Hashfunktion $h'(k) = 1 + (k \mod 5)$ benutzt werden.



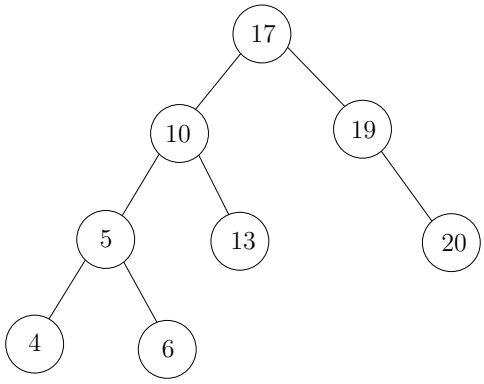
1 P f) Das untenstehende Array enthält die Elemente eines Min-Heaps in der üblichen Form gespeichert. Wie sieht das Array aus, nachdem das Minimum entfernt wurde und die Heap-Bedingung wieder hergestellt wurde?

1	5	2	6	9	7	12	10	8
1	2	3	4	5	6	7	8	9

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



1 P h) Fügen Sie in den untenstehenden AVL-Baum den Schlüssel 9 ein, und löschen Sie im entstandenen AVL-Baum den Schlüssel 17.



Nach Einfügen von 9:	Nach Löschen von 17:

1 P i) Gegeben sind sieben Schlüssel mit der jeweiligen Anzahl der Zugriffe. Erstellen Sie mit Hilfe des Huffman-Algorithmus einen optimalen Codierungsbaum, und zeichnen Sie diesen.

Schlüssel	a	b	c	d	e	f	g
Anzahl Zugriffe	6	2	4	3	13	3	12

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^3
- $n\sqrt{n}$
- $n!$
- $n \log n$
- $3^{\sqrt{n}}$
- $\dfrac{n^2}{\log n}$

3 P b) Gegeben ist die folgende Rekursionsgleichung:

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

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 4 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 := n until i < 2 loop
  from j := i until j < 1 loop
    j := j - 1
  end
  i := i // 2
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 // 8 loop
  from j := 2*i until j > n loop
    j := j + 1
  end
  i := i + 1
end
```

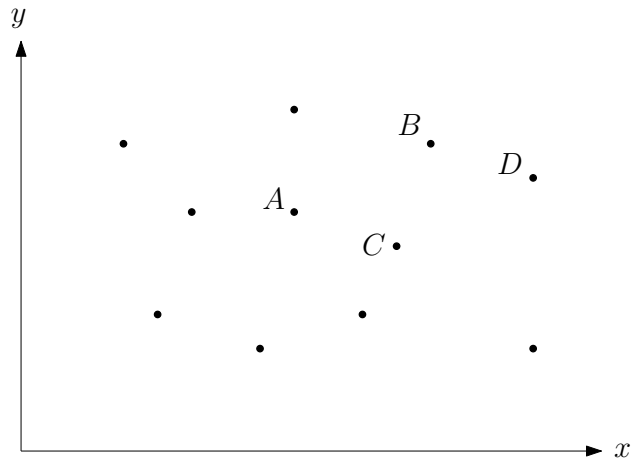
1 P

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 * k > n
    k := k + 1
  end
  i := i * 2
end
```

Aufgabe 3:

In dieser Aufgabe betrachten wir Mengen von Punkten in der Ebene. Wir sagen, dass ein Punkt (x, y) einen anderen Punkt (x', y') *dominiert*, wenn $x > x'$ und gleichzeitig $y > y'$ gilt. Ein Punkt heisst *dominant*, wenn er von keinem anderen Punkt dominiert wird.
Im untenstehenden Bild wird beispielsweise der Punkt A vom Punkt B dominiert, aber nicht vom Punkt C. Der Punkt D ist dominant.



3 P a) Entwerfen Sie einen möglichst effizienten Algorithmus, der für eine gegebene Menge von n Punkten eine Liste aller dominanten Punkte berechnet. Beschreiben Sie Ihren Algorithmus in Worten und/oder Pseudocode. Geben Sie zudem die Laufzeit Ihrer Lösung im schlechtesten Fall an.

6 P b) Wir betrachten nun eine dynamische Version des in a) gelösten Problems, bei der Punkte hinzugefügt werden können. Er können jedoch keine Punkte wieder gelöscht werden. Sie sollen also eine Datenstruktur entwerfen, welche folgende Operationen möglichst effizient (in Abhängigkeit der Anzahl an Punkten in der aktuellen Menge) unterstützt:

- `Init`(): erstellt eine Datenstruktur, welche eine leere Menge von Punkten repräsentiert
- `Insert`(x, y): fügt einen Punkt (x, y) zur aktuellen Menge von Punkten hinzu
- `ListEfficient`(): gibt die Liste aller dominanten Punkte in der aktuellen Menge aus

Beschreiben Sie Ihre Datenstruktur in Worten, und die Implementation der obigen drei Operationen in Worten und/oder Pseudocode.

Aufgabe 4:

In dieser Aufgabe betrachten wir ein zweidimensionales Array A mit n^2 verschiedenen Zahlen, bestehend aus n Zeilen und n Spalten. Zwei Elemente im Array sind *benachbart*, wenn sie sich an einer Seite berühren, d.h. das Element $A[i, j]$ ist mit den vier Elementen $A[i-1, j]$, $A[i, j-1]$, $A[i+1, j]$ und $A[i, j+1]$ benachbart, falls diese Felder existieren (Elemente am Rand des Arrays sind mit entsprechend weniger Elementen benachbart).
Eine Folge $x_1, x_2, \ldots, x_k$ von Elementen im Array heisst *Sequenz*, wenn folgende Bedingungen erfüllt sind:

- die Folge von Elementen ist aufsteigend sortiert, und
- für jedes $i \in 1, \ldots, k-1$ sind die Elemente x_i und x_{i+1} im Array benachbart.

3 P a) Entwerfen Sie einen möglichst effizienten Algorithmus, der im gegebenen Array eine längste Sequenz berechnet, die sich ganz in einer Zeile oder ganz in einer Spalte befindet. Im untenstehenden Beispiel-Array wäre eine mögliche Sequenz $4, 6, 10$. Diese ist jedoch keine längstmögliche Sequenz: Die Sequenz $6, 28, 29, 47$ ist länger. Beschreiben Sie Ihren Algorithmus in Worten, und geben Sie dessen Laufzeit an.

6 P b) Wir suchen nun eine längstmögliche beliebige Sequenz im gegebenen zweidimensionalen Array.

Im untenstehenden Beispiel-Array wäre eine mögliche Sequenz $4, 6, 28, 29, 47, 49$.

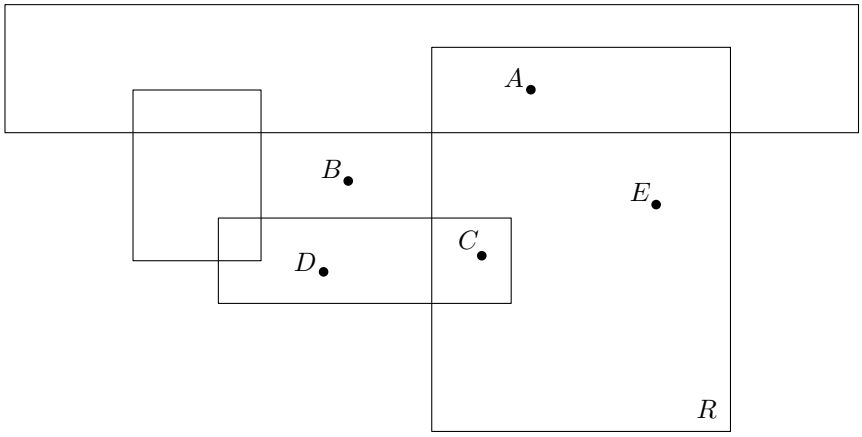
Entwerfen Sie einen möglichst effizienten Algorithmus, der eine solche längste Folge findet. Beschreiben Sie Ihren Algorithmus in Pseudocode, und geben Sie dessen Laufzeit an. Ihr Pseudocode muss in Java- Eiffel- oder C++-ähnlicher Notation geschrieben sein.

Beispiel-Array:

9	27	42	41	48
35	39	8	3	5
12	49	2	38	4
15	47	29	28	6
19	1	25	33	10

Aufgabe 5:

Gegeben sei eine Menge von Punkten in der Ebene, sowie eine Menge von iso-orientierten Rechtecken. Wir möchten bestimmen, ob es Punkte gibt, die innerhalb der Fläche liegen, welche von der Vereinigung aller Rechtecke überdeckt wird. Im unten stehenden Bild befinden sich z.B. die Punkte A, C, D und E innerhalb dieser Fläche, nicht jedoch der Punkt B.



Nehmen Sie im Folgenden an, dass als Input eine Menge von m iso-orientierten Rechtecken und eine Menge von n Punkten gegeben sind. Dabei ist jeder Punkt P_j, $j \in \{1, \ldots, n\}$, durch Koordinaten (x_j, y_j) und jedes Rechteck R_i, $i \in \{1, \ldots, m\}$, durch seine linke, obere Ecke (l_i, o_i) und seine rechte, untere Ecke (r_i, u_i), beschrieben. Sie können annehmen, dass alle Koordinaten ganzzahlig sind und im Wertebereich $\{1, \ldots, m + n\}$ liegen.

5 P a) Entwerfen Sie einen möglichst effizienten Algorithmus, der entscheidet (d.h. entweder "ja" oder "nein" ausgibt), ob sich (mindestens) einer der gegebenen Punkte innerhalb der von den Rechtecken überdeckten Fläche befindet. Beschreiben Sie Ihren Algorithmus in Worten und/oder in Pseudocode.

1 P b) Welche Laufzeit hat Ihr Verfahren im schlechtesten Fall, in Abhängigkeit von m und n?

3 P c) Nun möchten wir die maximale Anzahl von Punkten berechnen, die alle im gleichen Rechteck liegen. Beachten Sie dabei, dass ein Punkt in vielen Rechtecken gleichzeitig liegen kann. Im obigen Beispiel ist die gesuchte Anzahl gleich drei, da die Punkte A, C und E alle im Rechteck R liegen, und kein anderes Rechteck mehr Punkte enthält. Entwerfen Sie einen möglichst effizienten Algorithmus, der diese Anzahl bestimmt, und beschreiben Sie diesen in Worten und/oder in Pseudocode.