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. Januar 2012

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 Studierenden-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) Das Verfahren von Karatsuba/Ofman zur Multiplikation ganzer Zahlen berechnet das Produkt zweier Zahlen rekursiv anhand einer Formel, die bis auf Addition und Multiplikation mit der Basis (hier: 10) drei Produkte enthält. Geben Sie zwei Zahlen x und y an, für welche diese drei Produkte $(12 \cdot 34)$, $(56 \cdot 78)$ und $(12 \pm 78) \cdot (56 \pm 34)$ sind.

$x =$ _____, $y =$ _____

1 P b) Führen Sie auf dem gegebenen Array einen Aufteilungsschritt (in-situ, d.h. ohne Hilfsarray) des Sortieralgorithmus Quicksort durch. Benutzen Sie als Pivot das am rechten Ende stehende Element im Array.

25	12	83	2	58	68	19	34	47	99	37	56	41

1 P c) 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?

4	24	10	38	30	72	77	75	78
1	2	3	4	5	6	7	8	9

1 P d) Nehmen Sie an, die Schlüssel $1, 7, 42, 46, 35, 41, 18, 9, 30, 28, 31, 24, 19, 15, 27$ sollen in dieser Reihenfolge mittels Cuckoo-Hashing in Hashtabellen t_1 und t_2 eingefügt werden. Die Hashfunktion für Tabelle t_1 lautet $h_1(k) = k \mod 7$, die für Tabelle t_2 lautet $h_2(k) = 3k \mod 7$. Nehmen Sie ferner an, dass t_1 und t_2 jeweils die Grösse 7 haben. Wie lautet der erste Schlüssel in der oben genannten Folge, der nicht mehr eingefügt werden kann, ohne die Tabellen zu vergrössern (also eine `rehash` Operation auszuführen)?

Schlüssel: _____

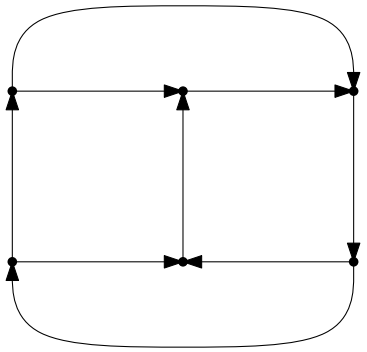
1 P e) Zeichnen Sie einen Baum mit einer ungeraden Anzahl von Kanten und einer minimalen Anzahl von Knoten, der ein Maximum Independent Set der Grösse 4 hat.

1 P f) Gegeben sind acht Schlüssel mit der relativen Anzahl der Zugriffe. Erstellen Sie mit Hilfe des Huffman-Algorithmus einen Codierungsbaum (Trie) und zeichnen Sie diesen.

Schlüssel	a	b	c	d	e	f	g	h
Anzahl Zugriffe	31	23	60	29	13	15	68	88

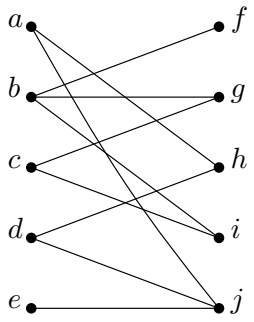
1 P

g) Markieren Sie in folgendem Graphen $G = (V, E)$ eine kleinstmögliche Menge S an Kanten, so dass der Graph $G' := (V, E \setminus S)$ eine topologische Sortierung hat :



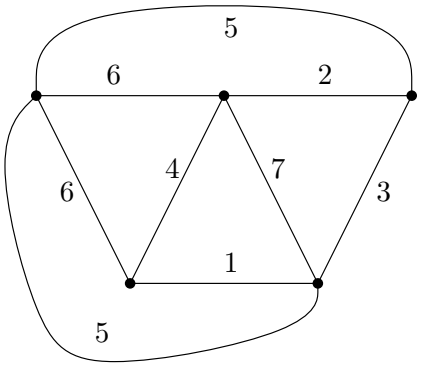
1 P

h) Geben Sie eine Teilmenge der Knoten des folgenden bipartiten Graphen an, die mit dem Satz von Hall beweist, dass der Graph kein perfektes Matching hat.



1 P

i) Welche Kante des folgenden Graphen wird als erste bei der Berechnung eines minimalen Spannbaums (MST) verworfen, wenn man das Verfahren von Kruskal verwendet?



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.

- $30\sqrt{n}$
- $3^{\frac{n}{2}}$
- $n!$
- $10n$
- $20\log(n)$
- $n\log n$
- $n^{\frac{2}{3}}$

3 P b) Gegeben ist die folgende Rekursionsgleichung:

$$T(n) := \begin{cases} 4 + 2T(\frac{n}{8}) & n > 1 \\ 1 & n = 1 \end{cases}$$

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 j := 1 until j > n loop
   from k := 2 until k > n loop
      k := k * 2
      from l := 1 until l > n loop
         l := l + 10
      end
   end
   j := j + 1
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 h := 1 until h > n loop
   from j := 1 until j > n*n loop
      j := j * 3
   end
   from k := 2 until k*k > n loop
      k := k + 1
   end
   h := h + 2
end
```

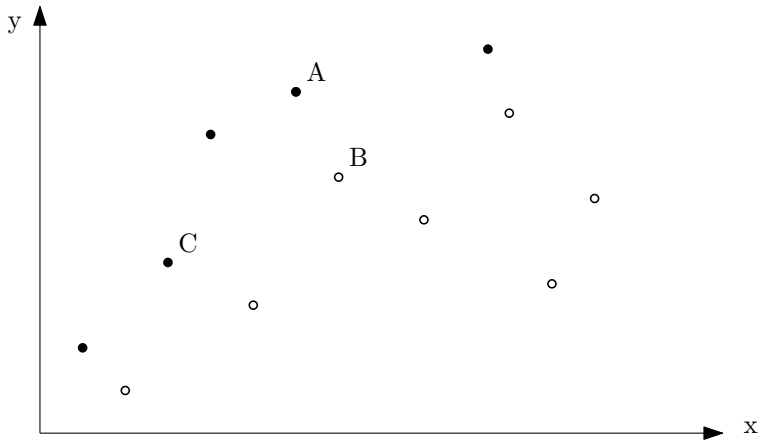
1 P

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

Aufgabe 3:

Ein Online-Shop für Smartphones möchte seinen Kunden nur die besten Geräte anbieten. Zwei wichtige Kriterien dabei sind der Preis und die maximale Betriebsdauer (des Akkus) eines Smartphones. Der Shop betreibt eine Datenbank, in der diese Informationen für jedes Smartphone hinterlegt werden. Den Kunden sollen nur die dominanten Smartphones vorgeschlagen werden: Wenn z.B. Gerät A zum Preis von 300CHF eine Betriebsdauer von 8h bietet und Gerät B zum Preis von 350CHF nur eine Betriebsdauer von 6h, so ist B nicht dominant.
Formal betrachten wir eine Menge von Punkten in der Ebene, wobei ein Punkt (x, y) ein Smartphone darstellt, wobei x den Preis und y die Betriebsdauer bezeichnet. Wir nehmen an, dass die Punkte in allgemeiner Lage liegen, d.h., keine zwei Punkte haben die gleiche x- oder y-Koordinate. 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 folgenden Bild dominiert der Punkt A den Punkt B. Punkt C dominiert weder A noch B. Die dominanten Punkte sind schwarz dargestellt.



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.

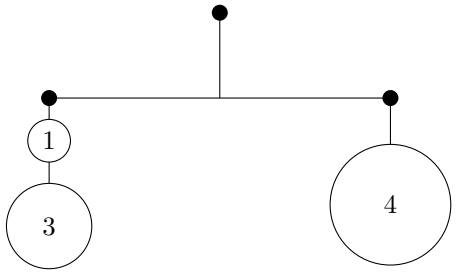
6 P b) Wir betrachten nun eine dynamische Version des in a) gelösten Problems, bei der Punkte hinzugefügt werden können. Er können jedoch keine Punkte wieder gelöscht werden. Sie sollen also eine Datenstruktur entwerfen, welche folgende Operationen möglichst effizient (in Abhängigkeit der Anzahl an Punkten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

Aufgabe 4:

Alexander (ein Künstler) plant ein Mobile zum Thema "Gleichgewicht" (siehe die Abbildung unten). Er hat n Scheiben zur Verfügung, welche zu beliebig langen Ketten aufgehangen werden können. Die Scheiben können unterschiedliche Gewichte haben: Scheibe i hat ein ganzzahliges Gewicht w_i. Die vorhandenen n Scheiben sollen auf die zwei Seiten (A und B) des Mobiles verteilt werden, so dass dieses im Gleichgewicht ist. Dies ist genau dann der Fall, wenn die Summe der Gewichte der Scheiben in A gleich der Summe der Gewichte der Scheiben in B ist. Zu beachten ist, dass *alle* n Scheiben benutzt werden müssen.



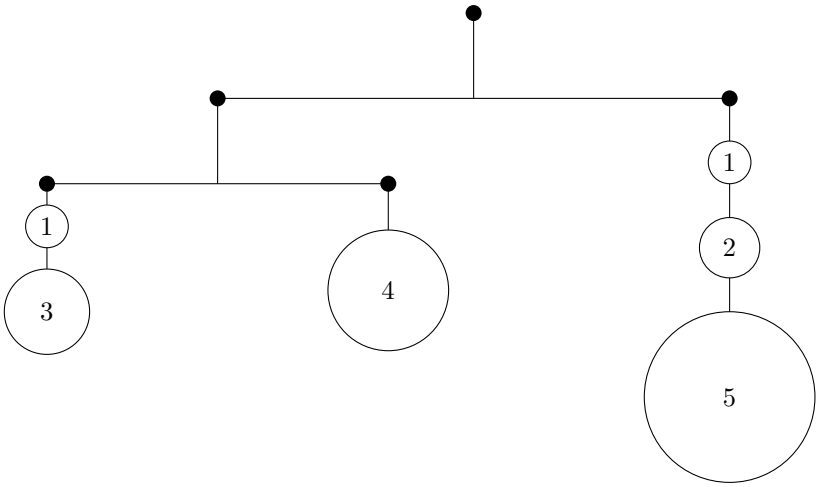
5 P a) Entwerfen Sie einen Algorithmus nach dem Muster der dynamischen Programmierung, der feststellt, ob für eine gegebene Menge von n Scheiben mit Gewichten $w_1, w_2, \ldots, w_n$ ein Gleichgewicht möglich ist. Schreiben Sie die Rekursionsgleichung des entsprechenden dynamischen Programms auf, und geben Sie zudem die Laufzeit Ihres Algorithmus an.

1 P b) Beschreiben Sie kurz in Worten, wie der Algorithmus in a) erweitert werden kann, um eine Aufteilung der Scheiben auszugeben, welche ein Gleichgewicht erreicht, falls eine solche existiert.

3 P c) Da die erste Skulptur ein Erfolg war, plant Alexander eine weitere (siehe die Abbildung unten). Für diese müssen die Scheiben in drei Mengen A, B und C aufgeteilt werden, so dass folgende Bedingungen gelten:

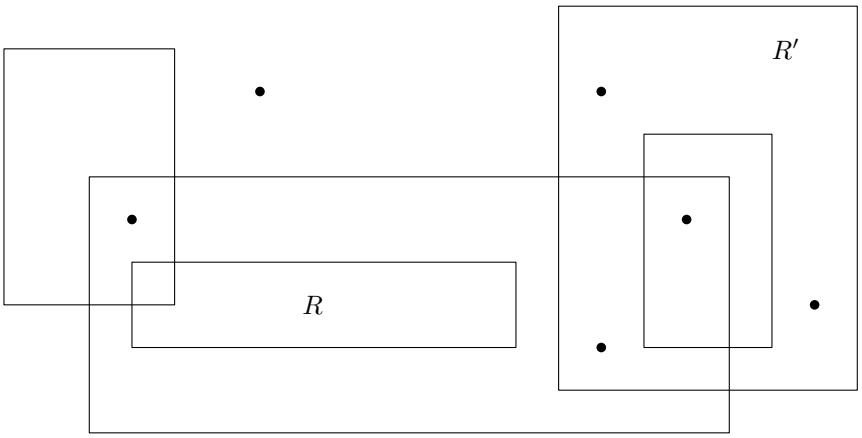
- Das Gesamtgewicht der Menge A ist gleich dem Gesamtgewicht der Menge B.
- Das Gesamtgewicht der Menge C ist gleich dem Gesamtgewicht der Menge A plus dem Gesamtgewicht der Menge B.

Wiederum müssen alle Scheiben verwendet werden. Entwerfen Sie ein dynamisches Programm, mit dem festgestellt werden kann, ob eine solche Aufteilung möglich ist, und geben Sie die Laufzeit Ihres Algorithmus an.



Aufgabe 5:

Gegeben sei eine Menge von Punkten in der Ebene, sowie eine Menge von iso-orientierten (achsenparallelen) Rechtecken. Wir möchten bestimmen, welche der Rechtecke keine der Punkte überdecken. Im unten stehenden Bild ist dies nur Rechteck R.



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 alle Rechtecke ausgibt, die keinen Punkt überdecken. Beschreiben Sie Ihren Algorithmus in Worten und/oder in Pseudocode.

2 P b) Welche Laufzeit hat Ihr Verfahren im schlechtesten Fall, in Abhängigkeit von m, n und der Anzahl der auszugebenden Rechtecke?

2 P c) Nun möchten wir ein Rechteck identifizieren, welches unter allen gegebenen Rechtecken die maximale Anzahl von Punkten überdeckt. Beachten Sie, dass es mehrere solche Rechtecke geben kann. Beachten Sie auch, dass ein Punkt in vielen Rechtecken gleichzeitig liegen kann. Im obigen Beispiel ist das gesuchte Rechteck R', da kein anderes Rechteck mehr Punkte enthält. Entwerfen Sie einen möglichst effizienten Algorithmus, der ein solches Rechteck ausgibt, und beschreiben Sie diesen in Worten und/oder in Pseudocode.