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
Tobias Pröger

Prüfung
Datenstrukturen und Algorithmen
D-INFK

8. August 2013

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.
- Sie dürfen alle Algorithmen und Datenstrukturen aus der Vorlesung verwenden, ohne sie noch einmal zu beschreiben. Wenn Sie sie modifizieren, reicht es, die Modifikationen zu beschreiben.
- Die Prüfung dauert 180 Minuten.

Viel Erfolg!

Stud.-Nummer: _____

Aufgabe	1	2	3	4	5	**Σ**
Mögl. Punkte	8	10	13	9	10	50
$\sum$ 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 einen Aufteilungsschritt (in-situ, d.h. ohne Hilfsarray) des Sortieralgorithmus *Quicksort* durch. Benutzen Sie als Pivot das am rechten Ende stehende Element im Array.

9	5	19	11	7	15	1	0	2	17	4	8
1	2	3	4	5	6	7	8	9	10	11	12

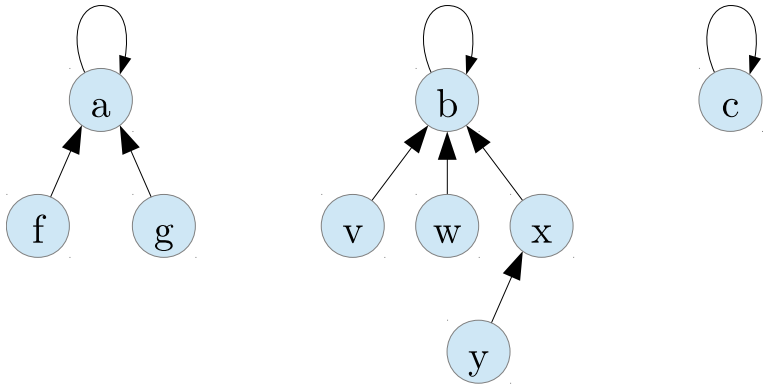
1	2	3	4	5	6	7	8	9	10	11	12

1 P (b) Zeichnen Sie einen AVL-Baum mit 7 Knoten, bei dem *jeder* innere Knoten einen Balancierungsfaktor ungleich 0 besitzt.

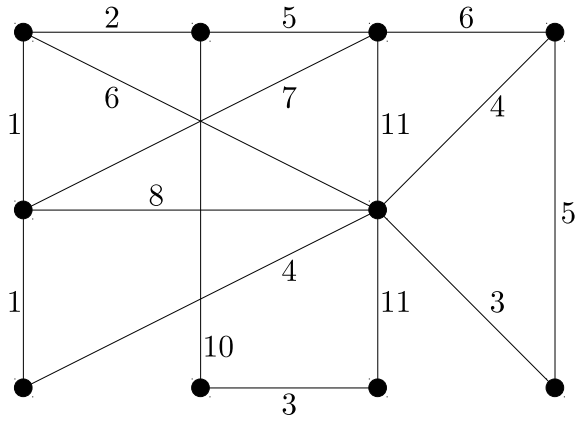
3 P (c) Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind. Jede korrekte Antwort gibt 0,5 Punkte, für jede falsche Antwort werden 0,5 Punkte abgezogen. Eine fehlende Antwort gibt 0 Punkte. Insgesamt gibt die Aufgabe mindestens 0 Punkte. Sie müssen Ihre Antworten nicht begründen.

Eine Inorder-Traversierung eines binären Suchbaums erzeugt eine sortierte Liste der gespeicherten Schlüssel.	☐ Wahr	☐ Falsch
Hat eine Folge von m Operationen im schlimmsten Fall Gesamtkosten $\mathcal{O}(m)$, dann hat jede einzelne dieser Operationen im schlimmsten Fall Kosten $\mathcal{O}(1)$.	☐ Wahr	☐ Falsch
Sei $G = (V, E)$ ein Graph. Jeder Teilgraph von G mit $\|V\|-1$ Kanten ist ein Spannbaum von G.	☐ Wahr	☐ Falsch
In einem Fibonacci-Heap mit n Schlüsseln ist die Laufzeit zur Extraktion des Minimums im schlimmsten Fall $\mathcal{O}(\log n)$.	☐ Wahr	☐ Falsch
Sortieren durch Auswahl ist ein stabiles Sortierverfahren.	☐ Wahr	☐ Falsch
Auf einem komplett vorsortierten Array ist die Laufzeit von Sortieren durch Auswahl linear.	☐ Wahr	☐ Falsch

1 P (d) Führen Sie auf der folgenden Union-Find-Datenstruktur zuächst Union(a, c) und danach Union(Find$(f), b)$ aus. Benutzen Sie das Verfahren "Vereinigung nach Höhe", und zeichnen Sie die nach diesen zwei Operationen resultierende Datenstruktur.



1 P (e) Markieren Sie im untenstehenden gewichteten Graphen die erste Kante, die der Algorithmus von Kruskal *nicht* in den minimalen Spannbaum aufnimmt.



1 P (f) Zeichnen Sie den binären Suchbaum zur Schlüsselmenge $\{1, \ldots, 8\}$, dessen Preorder-Reihenfolge mit $3, 2, 1, 5, 4$ beginnt, und dessen Postorder-Reihenfolge mit $7, 8, 6, 5, 3$ endet.

Aufgabe 2.

1 P (a) Geben Sie für die untenstehenden Funktionen eine **Reihenfolge** an, so dass folgendes gilt: Wenn eine Funktion f links von einer Funktion g steht, dann gilt $f \in \mathcal{O}(g)$.

Beispiel: Die drei Funktionen n^3, n^7, n^9 sind bereits in der entsprechenden Reihenfolge, da $n^3 \in \mathcal{O}(n^7)$ und $n^7 \in \mathcal{O}(n^9)$ gilt.

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

3 P (b) Gegeben ist die folgende Rekursionsgleichung:

$$T(n) := \begin{cases} 6 + 7T(n/3) & n > 1 \\ 6 & n = 1 \end{cases}$$

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 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 $n \in \mathbb{N}$ für den folgenden Algorithmus (so knapp wie möglich) in Θ-Notation an. Sie müssen Ihre Antwort nicht begründen.

```
1 for(int i = 1; i <= n*n; i += 10) {
2         for(int j = 1; j*j <= n; j ++)
3                 ;
4 }
```

1 P (d) Geben Sie die asymptotische Laufzeit in Abhängigkeit von $n \in \mathbb{N}$ für den folgenden Algorithmus (so knapp wie möglich) in Θ-Notation an. Sie müssen Ihre Antwort nicht begründen.

```
1 for(int i = n; i > 1; i = i/2) {
2         for(int j = 1; j <= n; j ++)
3                 ;
4 }
```

4 P (e) Wir betrachten einen Dezimalzähler, der mit 0 initialisiert wird und nur eine einzige Operation ERHÖHE unterstützt, die den Wert des Zählers um 1 erhöht. Die Kosten für diese Operation entsprechen genau der Anzahl der Stellen, die verändert werden. Hat der Zähler z.B. den Wert 18 und wird ERHÖHE aufgerufen, dann ist der neue Wert des Zählers 19, und die Kosten für die Operation betragen 1 (nur die 8 wurde verändert). Hat der Zähler den Wert 19999 und wird ERHÖHE aufgerufen, dann ist der Wert des Zählers 20000, und die Operation hatte Kosten 5.

Zeigen Sie nun mittels amortisierter Analyse, dass die Kosten der Operation ERHÖHE amortisiert $\mathcal{O}(1)$ sind. Definieren Sie dazu eine geeignete Kontostandsfunktion Φ_i, und geben Sie jeweils die realen und die amortisierten Kosten an, wenn die letzten $k \geq 0$ Stellen des Zählers den Wert 9 haben.

Aufgabe 3.

Ein *Palindrom* ist eine Zeichenkette, die sich von vorne wie von hinten gleich liest, also z.B. das Wort `RENTNER`. Formal ist ein Palindrom eine Zeichenkette $\langle a_1, \ldots, a_n \rangle$, wobei entweder $n = 1$ gilt, oder aber es sind $a_1 = a_n$ und $\langle a_2, \ldots, a_{n-1} \rangle$ ein Palindrom. Ein Array $A[1 \ldots n]$ speichere eine Zeichenkette der Länge n. Ein Teilarray $A[i \ldots j]$, $1 \leq i \leq j \leq n$, heisst *Palindrom in* A, falls $\langle A[i], \ldots, A[j] \rangle$ ein Palindrom ist.

Beispiel: Das Array [`L`, `A`, `R`, `A`] enthält die Palindrome `A`, `R`, `L` sowie `ARA` (das Palindrom `A` kommt doppelt vor). Das Array [`A`, `N`, `N`, `A`] enthält die Palindrome `A`, `N`, `NN` sowie `ANNA` (die Palindrome `A` und `N` kommen doppelt vor).

9 P (a) Sei A ein Array, das eine Zeichenkette der Länge n speichert. Entwerfen Sie einen Algorithmus nach dem Prinzip der *dynamischen Programmierung*, der alle Paare (i, j) ausgibt, für die $\langle A[i], \ldots, A[j] \rangle$ ein Palindrom ist. Gehen Sie in Ihrer Lösung auf die folgenden Aspekte ein.

1) Was ist die Bedeutung eines Tabelleneintrags, und welche Grösse hat die DP-Tabelle?
2) Wie berechnet sich ein Tabelleneintrag aus früher berechneten Einträgen? Unterscheiden Sie bei der Beschreibung Palindrome der Längen 1, 2 sowie längere Palindrome.
3) In welcher Reihenfolge müssen die Einträge berechnet werden?
4) Wie lässt sich die Lösung aus der DP-Tabelle extrahieren?

Beispiel: Für die Eingabe [`L`, `A`, `R`, `A`] werden die Paare $(1, 1)$, $(2, 2)$, $(3, 3)$, $(4, 4)$, $(2, 4)$ ausgegeben. Für die Eingabe [`A`, `N`, `N`, `A`] werden die Paare $(1, 1)$, $(2, 2)$, $(3, 3)$, $(4, 4)$, $(2, 3)$, $(1, 4)$ ausgegeben. Es wird keine spezielle Reihenfolge gefordert, in der die Paare ausgegeben werden müssen.

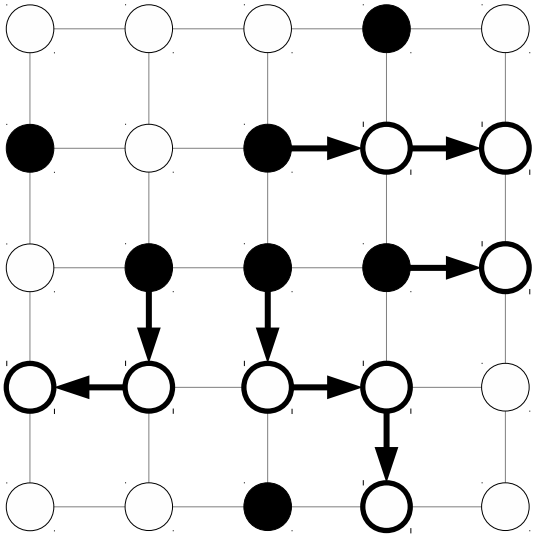
Beachten Sie, dass die Aufgabe ohne dynamische Programmierung durch triviale Aufzählung aller Palindrome in Zeit $\mathcal{O}(n^3)$ lösbar ist. Gesucht wird hier ein effizienterer Algorithmus.

1 P (b) Geben Sie die Laufzeit Ihrer Lösung an.

3 P (c) Angenommen, der Algorithmus aus (a) hat die DP-Tabelle bereits berechnet. Beschreiben Sie detailliert, wie Sie aus der DP-Tabelle ein längstes Palindrom in A ablesen können. Geben Sie auch die maximal benötigte Laufzeit an.

Aufgabe 4.

Gegeben sei ein $n \times n$-Gitter, d.h. ein ungerichteter Graph mit n Zeilen und n Spalten, der Kanten zwischen je zwei horizontal sowie vertikal benachbarten Knoten enthält. Beim *Fluchtwegeproblem* sind $m \leq n^2$ Startknoten $s_1, \ldots, s_m$ gegeben. Wir möchten entscheiden, ob es m kantendisjunkte Wege von den Startpunkten s_i zu je einem Randpunkt des Gitters gibt. Beachten Sie, dass je zwei Wege zwar einen oder mehrere Knoten, aber keine gemeinsamen Kanten besitzen dürfen. Im folgenden Beispiel sind die Startknoten schwarz und die Fluchtwege fett eingezeichnet.



4 P (a) Modellieren Sie das obige Entscheidungsproblem als Flussproblem. Beachten Sie, dass wir bei Flussproblemen *gerichtete* Netzwerke betrachten, das Gitter selbst aber ungerichtet ist. Erläutern Sie daher genau, wie ein geeignetes Netzwerk $N = (V, E, c)$ konstruiert werden kann, d.h. welche Knoten V und welche Kanten E definiert werden, und welche Kapazitäten den Kanten zugewiesen werden müssen. Überlegen Sie, wie Sie aus dem Wert eines maximalen Flusses schlussfolgern können, ob m kantendisjunkte Fluchtwege existieren oder nicht.

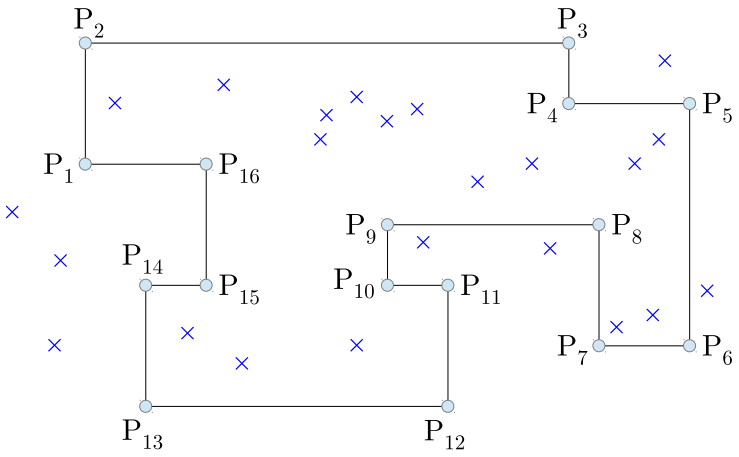
2 P (b) Nennen Sie einen Flussalgorithmus, der das Problem aus (a) möglichst effizient löst. Geben Sie die Laufzeit der Lösung aus (a) in Abhängigkeit von n und m an.

3 P (c) Wir möchten nun entscheiden, ob es eine Menge von m *knotendisjunkten* Fluchtwegen gibt, d.h. eine Menge von Fluchtwegen, von denen keine zwei einen gemeinsamen Knoten besitzen. Beschreiben Sie detailliert, wie Sie Ihre Lösung aus (a) modifizieren müssen, damit durch jeden Knoten maximal ein Fluchtweg läuft. Verändert sich die Laufzeit Ihrer Lösung, und wenn ja, wie?

Aufgabe 5.

Wir betrachten eine Menge von n Kühen sowie ein Gehege aus m geradlinigen orthogonalen Zaunsegmenten. Die Position jeder Kuh i, $1 \leq i \leq n$, kann mittels eines GPS-Empfängers bestimmt werden und ist durch den Punkt $K_i \in \mathbb{Q}^2$ gegeben. Das Gehege ist ein geschlossenes einfaches Polygon mit den Eckpunkten (also den Zaunpfählen) $P_i \in \mathbb{Q}^2$, $1 \leq j \leq m$. Dabei seien P_i und P_{i+1} für $i \in \{1, \ldots, m-1\}$ und zusätzlich P_m und P_1 jeweils durch ein Zaunsegment verbunden. Weiterhin sei P_1 der am weitesten links liegende Punkt (falls es mehr als nur einen solchen Punkt gibt, dann sei P_1 unter diesen der am weitesten unten liegende Punkt). Sie dürfen davon ausgehen, dass sich keine Kuh direkt auf einem Zaunsegment befindet, d.h. kein Punkt K_i liegt auf einer Polygonkante. Wir wollen ermitteln, wie viele Kühe sich innerhalb des Geheges befinden.

9 P (a) Wir nehmen der Einfachheit halber an, dass alle Zaunsegmente parallel zur x- oder zur y-Achse verlaufen. Geben Sie einen Scanline-Algorithmus an, der als Eingabe die Positionen der Kühe K_i sowie die Positionen der Zaunpfähle P_j erhält, und die Anzahl der Kühe berechnet, die sich innerhalb des Geheges befinden.



1 P (b) Geben Sie die Laufzeit des Verfahrens aus (a) an.