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
Markus Püschel
Thomas Tschager
Tobias Pröger

Beispielprüfung
Datenstrukturen und Algorithmen
D-INFK

15. Dezember 2016

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 120 Minuten.

Viel Erfolg!

Stud.-Nummer: _____

Aufgabe	1	2	3	**Σ**
Mögl. Punkte	14	6	13	33
$\sum$ Punkte				

Aufgabe 1. **/ 14 P**

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) Fügen Sie die Schlüssel 4, 16, 20, 6, 12, 9, 5 in dieser Reihenfolge in die untenstehende Hashtabelle ein. Benutzen Sie offenes Hashing mit der Hashfunktion $h(k) = k \bmod 11$. Lösen Sie Kollisionen mittels quadratischem Sondieren auf. Im Falle einer Kollision soll die Sondierung zunächst nach links und erst danach nach rechts erfolgen.

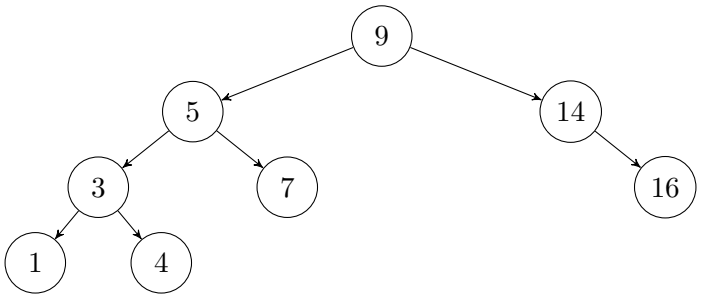
0	1	2	3	4	5	6	7	8	9	10

/ 1 P b) Gegeben sei die Schlüsselmenge $\mathcal{K} = \{5, 9, 8, 11, 15, 7, 20\}$. Zeichnen Sie die beiden binären Suchbäume, die genau die Schlüssel aus $\mathcal{K}$ verwalten und die unter allen möglichen Suchbäumen minimale bzw. maximale Höhe haben.

Baum minimaler Höhe:	Baum maximaler Höhe:

/ 1 P

c) Fügen Sie in den folgenden AVL-Baum *zuerst* den Schlüssel 2 ein und entfernen Sie *danach* den Schlüssel 14.



Nach Einfügen von 2:	Nach Löschen von 14:

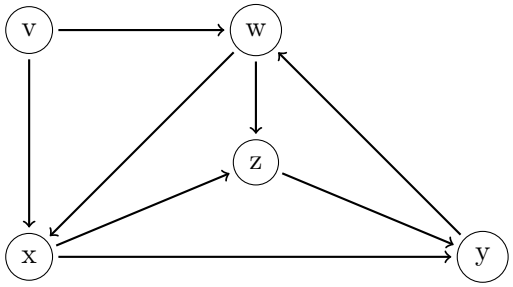
/ 2 P

d) 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
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

e) Gegeben Sei der folgende Graph $G = (V, E)$. Geben Sie eine Menge $E' \subset E$ kleinstmöglicher Kardinalität an, sodass $G' = (V, E \backslash E')$ topologisch sortiert werden kann. Geben Sie ausserdem eine topologische Sortierung für G' an.



$E' =$ _____

topologische Sortierung für G':

_____

/ 1 P

f) 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.

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

1	2	3	4	5	6	7	8	9	10	11

/ 1 P

g) Das folgende Array enthält die Elemente eines in üblicher Form gespeicherten Min-Heaps. Entfernen Sie das minimale Element aus dem Heap, stellen Sie die Heapbedingung wiederher, und geben Sie das resultierende Array an.

2	8	10	9	12	15	11	13	14
1	2	3	4	5	6	7	8	9

1	2	3	4	5	6	7	8

/ 3 P h) Gegeben ist die folgende Rekursionsgleichung:

$$T(n) := \begin{cases} T(n/5) + 4n + 1 & n > 1 \\ 5 & n = 1 \end{cases}$$

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 5 ist.

(2) Für $q \neq 1$ gilt: $\sum_{i=0}^{k} q^i = \frac{q^{k+1}-1}{q-1}$.

Herleitung (falls benötigt):

Geschlossene und vereinfachte Formel:

$T(n) = T(5^k) =$

Induktionsbeweis:

/ 1 P

i) 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/2; i += 2)
2         for(int j = n; j >= i; j -= 1)
3                 for(int k = n; k > 2; k /= 2)
4                         ;
```

/ 1 P

j) 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 ++) {
2         for(int j = 1; j <= i; j *= 2)
3                 ;
4         for(int k = 1; k*k <= n; k += 1)
5                 ;
6 }
```

/ 1 P

k) 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.

$$\sqrt{n^3},\ \binom{n}{4},\ n!,\ \frac{n^2}{\log n},\ n \log n,\ 3^n,\ (\log n)^4$$

Lösung: _____, _____, _____, _____, _____, _____, _____.

Aufgabe 2. **/ 6 P**

Im Devisenhandel bezeichnet *Arbitrage* das Ausnutzen von Preisunterschieden, um durch mehrfaches Wechseln von Währungen einen Gewinn zu erzielen. Am 2. Juni 2009 zum Beispiel konnte 1 US-Dollar in 95.729 Yen gewechselt werden, 1 Yen in 0.00638 Britische Pfund und 1 Britisches Pfund in 1.65133 US-Dollar. Hätte ein Händler also 1 US-Dollar in Yen gewechselt, den erhaltenen Betrag in Britische Pfund und schliesslich diesen Betrag dann zurück in US-Dollar getauscht, dann hätte er $95.729 \cdot 0.00638 \cdot 1.65133 \approx 1.0086$ US-Dollar erhalten, was einem Gewinn von 0.86% entspricht.

/ 3 P a) Gegeben seien n Währungen $\{1, \ldots, n\}$ und eine $(n \times n)$-Wechselkursmatrix $R \in (\mathbb{Q}^+)^2$. Für zwei Währungen $i, j \in \{1, \ldots, n\}$ kann eine Einheit der Währung i in $R(i, j) > 0$ Einheiten der Währung j getauscht werden. Es soll entschieden werden, ob ein Arbitrage-Geschäft möglich ist, d.h., ob es eine Folge von k verschiedenen Währungen $W_1, \ldots, W_k \in \{1, \ldots, n\}$ mit $R(W_1, W_2) \cdot R(W_2, W_3) \cdots R(W_{k-1}, W_k) \cdot R(W_k, W_1) > 1$ gibt.

Modellieren Sie das Problem als Graphproblem. Konstruieren Sie aus der obigen Eingabe einen gerichteten, gewichteten Graphen $G = (V, E, w)$, der *genau dann* einen Kreis mit negativem Gewicht besitzt, wenn ein Arbitrage-Geschäft möglich ist. Begründen Sie, dass G genau dann einen Kreis negativer Länge enthält, wenn ein Arbitrage-Geschäft möglich ist.

Hinweis: Die Benutzung des Logarithmus kann sinnvoll sein, denn es gilt $\ln(a \cdot b) = \ln(a) + \ln(b)$.

/ 3 P b) Welcher Kürzeste-Wege-Algorithmus kann benutzt werden, um in einem Graphen Kreise negativer Länge zu erkennen? An welchem Knoten kann der Algorithmus beginnen? Welche Laufzeit (in Abhängigkeit von n) erreicht der gewählte Algorithmus, wenn er auf den Graphen in a) angewendet wird?

Aufgabe 3. **/ 13 P**

Eine Firma hat den Auftrag bekommen, ein Kabel der Länge mindestens L zu produzieren. Da es im Lager viele bereits früher produzierte Kabelstücke gibt, soll das gewünschte Kabel aus diesen Teilstücken zusammengesetzt werden. Konkret gibt es im Lager n Kabelstücke. Kabelstück i hat Länge l_i. Werden zwei Kabelstücke mit Längen l_i und l_j zusammengesetzt, entsteht ein Kabel der Länge $l_i + l_j$. Um das entstehende Kabel nicht unnötig lang zu machen, soll es unter allen Möglichkeiten, ein Kabel der Länge $\geq L$ herzustellen, minimale Länge haben. Sie dürfen annehmen, dass der Lagerbestand ausreichend gross ist, d.h., dass die Summe der Längen aller Kabelstücke mindestens L beträgt.

Beispiel: Es soll ein Kabel der Länge $L = 6$ produziert werden, und im Lager gibt es Kabelstücke der Längen $l_1 = 3$, $l_2 = 4$ und $l_3 = 5$. Die beste Möglichkeit ist die Wahl der Kabelstücke $\{1, 2\}$, denn diese haben zusammen Länge 7. Auch die Wahlen $\{2, 3\}$ und $\{1, 2, 3\}$ führen zu einem Kabel der Länge ≥ 6, aber da ihre Gesamtlängen 9 bzw. 12 betragen, sind sie nicht optimal und daher auch nicht die gesuchte Lösung.

/ 9 P a) Geben Sie einen Algorithmus an, der nach dem Prinzip der dynamischen Programmierung arbeitet und die minimale Länge eines Kabels berechnet, sodass dieses Kabel aus geeigneten Kabelstücken aus $\{1, \ldots, n\}$ zusammengesetzt werden kann und mindestens Länge L hat. Für das Beispiel oben soll also 7 ausgegeben werden. 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?
3) In welcher Reihenfolge können die Einträge berechnet werden?
4) Wie kann aus der DP-Tabelle der Wert der minimalen Kabellänge ausgelesen werden?

Hinweis: Der triviale Algorithmus, der einfach alle möglichen Lösungen inspiziert, gibt **keine** Punkte, weil er nicht nach dem Prinzip der dynamischen Programmierung arbeitet.

/ 2 P b) Beschreiben Sie detailliert, wie aus der DP-Tabelle abgelesen werden kann, welches Kabelstück in einer optimalen Lösung benutzt wird.

/ 2 P c) Geben Sie die Laufzeit des in a) und b) entwickelten Verfahrens an und begründen Sie Ihre Antwort. Ist die Laufzeit polynomiell?