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

9. Feburar 2011

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) 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.

52	22	39	5	10	86	6	53	17	74	63	72	38

1 P b) Nehmen Sie an, die Schlüssel $7, 12, 33, 35, 14, 28, 36, 39, 47, 8, 46, 42, 16, 48, 25$ 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 in eingefügt werden kann, ohne die Tabellen zu vergrössern (also eine `rehash` Operation auszuführen)?

Schlüssel: _____

1 P c) Zeichnen Sie den binären Suchbaum, dessen Postorder-Traversierung die Folge $5, 25, 24, 54, 39, 8, 57, 55$ ergibt.

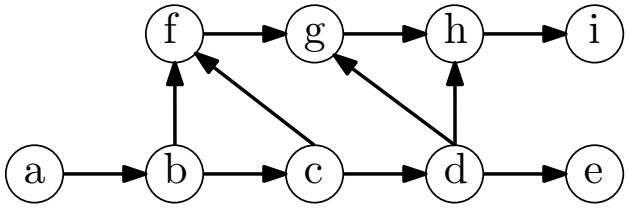
1 P

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?

9	15	13	52	38	70	64	78	73
1	2	3	4	5	6	7	8	9

1 P

e) 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.

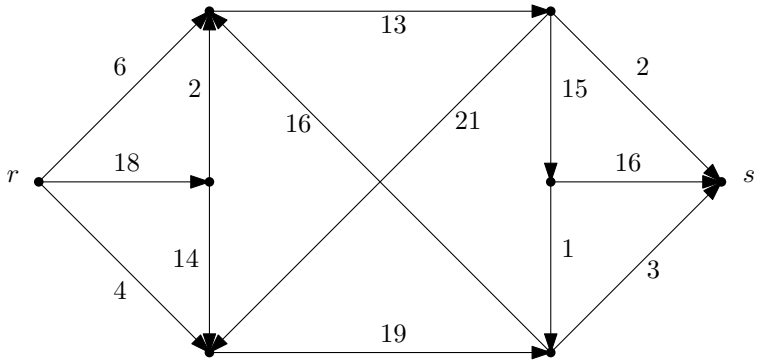


1 P

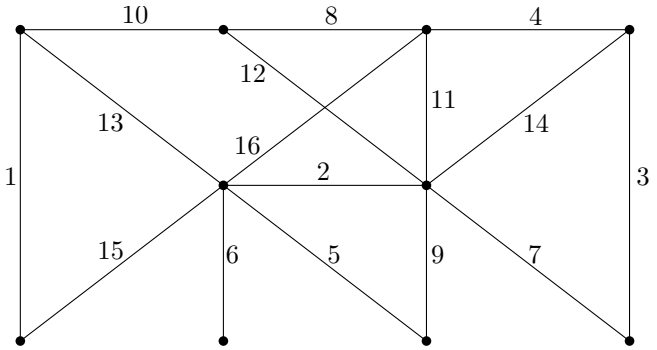
f) Gegeben sind acht Schlüssel mit der relativen Anzahl der Zugriffe. Erstellen Sie mit Hilfe des Huffman-Algorithmus einen Codierungsbaum (Trie).

Schlüssel	a	b	c	d	e	f	g	h
Anzahl Zugriffe	72	32	60	74	22	15	29	64

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) Markieren Sie im untenstehenden gewichteten Graphen die erste Kante, die vom Algorithmus von Kruskal zur Berechnung eines minimalen Spannbaums *nicht* in den Spannbaum aufgenommen wird.



1 P i) Sei T der Splay-Tree, dessen Preorder-Traversierung die Werte $7, 6, 1, 4, 2, 5, 8, 9$ liefert. Zeichnen Sie den Splay-Tree, der entsteht, nachdem man den Knoten 6 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.

- $\sqrt{2^n}$
- $\sqrt{n}$
- $\frac{1}{n!}$
- $\log(n)$
- n^2
- $\log(n^n)$

3 P b) Gegeben ist die folgende Rekursionsgleichung:

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

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 2 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*n loop
  from j := 1 until j > n 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 := n until j < i loop
    j := j - 1
    from k := 1 until k > n loop
      k := k + 10
    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 j := 1 until j*j > n loop
    j := j + 1
  end
  from k := 1 until k > n loop
    k := k + i
  end
  i := i + 1
end
```

Aufgabe 3:
Sie wurden beauftragt, ein Theaterfestspiel zu planen. Gegeben ist eine Menge $V = \{v_1, \cdots, v_n\}$ von Vorstellungen, aus der eine Teilmenge wie folgt ausgewählt werden soll. Es steht genau eine Bühne zur Verfügung. Jede Vorstellung hat eine Startzeit s_i und eine Endzeit e_i, wobei $0 \leq s_i < e_i < \infty$. Sofern eine Vorstellung v_i eingeplant ist, besetzt sie die Bühne im Zeitintervall $[s_i, e_i)$ (auf der rechten Seite offen). Keine zwei Vorstellungen aus V besetzen genau das gleiche Intervall. Zwei Vorstellungen v_i, v_j nennt man *kompatibel*, falls sich ihre Intervalle $[s_i, e_i)$ und $[s_j, e_j)$ nicht überlappen, d.h., wenn entweder $e_i \leq s_j$ oder $e_j \leq s_i$ gilt. Ihre Aufgabe ist es, einen Plan zu erstellen, so dass möglichst viele zueinander kompatible Vorstellungen stattfinden.

Beispiel: Bei einer Menge $V = \{v_1, v_2, v_3\}$, mit $s_1 = 10, e_1 = 13; s_2 = 11, e_2 = 14;\ s_3 = 13,\ e_3 = 16$, bilden die Vorstellungen $\{v_1, v_3\}$ die maximale Menge von kompatiblen Vorstellungen.

Hinweise: Man beachte, dass zwei mögliche Vorstellungen v_k und v_ℓ auch dann noch zueinander kompatibel sind, wenn die Endzeit der einen gleich der Startzeit der anderen ist, d.h. falls $e_k = s_\ell$.

3 P a) Beschreiben Sie kurz in Worten einen Ansatz für einen Algorithmus, der aus einer gegebenen Menge V von Vorstellungen möglichst effizient eine maximale Menge von zueinander kompatiblen Vorstellungen berechnet.

5 P b) Schreiben Sie in Pseudocode eine möglichst effiziente Funktion, die, ausgehend von einer Menge V, eine maximale Menge von zueinander kompatiblen Vorstellungen nach dem in Teilaufgabe a) angegebenen Algorithmus berechnet.

1 P c) Geben Sie die Laufzeit der in b) implementierten Funktion an.

Aufgabe 4:
Die Marketingabteilung einer Bank möchte Daten ihres "Kunden werben Kunden"-Programms auswerten. Jeder Kunde der Bank ist entweder direkt durch die Bank angeworben worden, oder durch einen schon bestehenden Kunden. Die Bank interessiert sich nun dafür, ob ein Kunde a von einem Kunden b direkt oder indirekt geworben wurde.

Formal soll folgendes Problem gelöst werden: Gegeben sei eine Menge $K = \{1, \ldots, n\}$ von Kunden sowie eine Menge S von geordneten Paaren (u, v), $u \in (K \cup \{0\})$, $v \in K$, $u \neq v$, mit der Bedeutung, dass Werber u (wobei 0 die Bank repräsentiert) den Kunden v angeworben hat. Für jeden Kunden $v \in K$ gibt es genau ein Paar $(u, v) \in S$, da der Werber für jeden Kunden eindeutig ist. Wir sagen, dass v ein *Nachfolger* von u ist, falls es ein k und eine Folge von Kunden $u = v_0, v_1, v_2, \ldots, v_k = v$ gibt, so dass $(v_i, v_{i+1}) \in S$ für alle $0 \leq i < k$. Die Operation `nachfolger(`a`,`b`)` liefert den Wert `true` zurück, wenn a Nachfolger von b ist, ansonsten `false`.

Entwickeln Sie für die folgenden Fälle jeweils eine Datenstruktur, welche die geforderten Operationen möglichst effizient unterstützt.

Hinweis: Sie dürfen aus der Vorlesung bekannte Algorithmen verwenden, ohne deren Pseudocode hier zu wiederholen.

4 P a) Neben der Initialisierung der Datenstruktur durch gegebene Mengen K und S soll zunächst nur die Operation `nachfolger(`a`,`b`)` unterstützt werden. Insbesondere sollen K und S statisch sein, d.h., dass weder Einfüge- noch Löschoperationen gestattet sind.

Beschreiben Sie Ihre Datenstruktur in Worten. Geben Sie die zur Initialisierung der Datenstruktur benötigte Laufzeit in Abhängigkeit von K und S an. Geben Sie die Operation `nachfolger(`a`,`b`)` in Pseudocode an, sowie deren Laufzeit.

2 P b) Nehmen Sie nun an, dass zusätzlich zu a) eine Operation `loesche(`u`)` unterstützt werden soll, die es erlaubt, einen Kunden $u \in K$ zu entfernen, falls u keinen Nachfolger hat, und die ansonsten eine Fehlermeldung ausgibt.

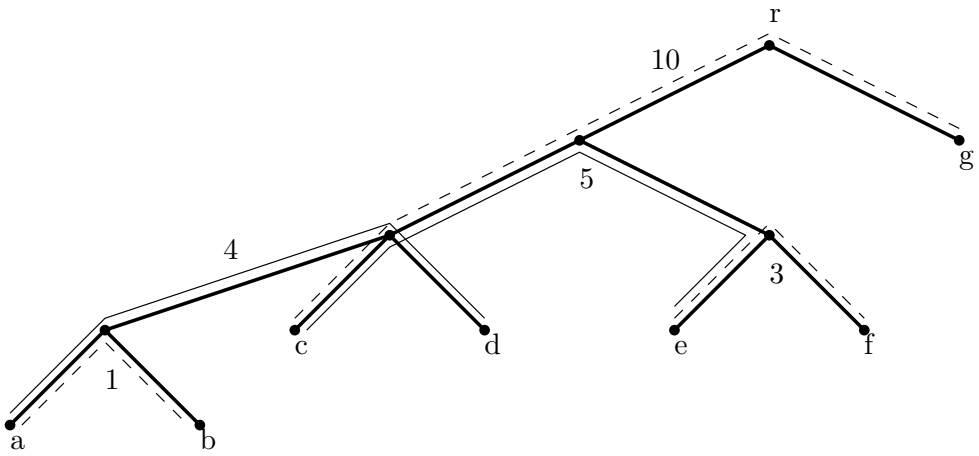
Geben Sie die Operation `loesche(`u`)` in Pseudocode an, sowie deren Laufzeit. Beschreiben Sie etwaige Änderungen Ihrer in a) entwickelten Datenstruktur in Worten (oder geben Sie eine neue Datenstruktur an). Geben Sie auch etwaige Änderungen der unter a) gefragten Laufzeiten an.

3 P c) Schliesslich soll auch eine Operation `geworben(`a`,`b`)` unterstützt werden, die es erlaubt, einen neuen Kunden $a \notin K$ als Nachfolger eines Kunden $b \in K$ einzufügen.

Geben Sie die Operation `geworben(`a`,`b`)` in Pseudocode an, sowie deren Laufzeit. Beschreiben Sie etwaige Änderungen Ihrer in b) entwickelten Datenstruktur in Worten (oder geben Sie eine neue Datenstruktur an). Geben Sie auch etwaige Änderungen der unter a) und b) gefragten Laufzeiten an.

Hinweis: In dieser Teilaufgabe dürfen Sie annehmen, dass für rationale Zahlen beliebiger Genauigkeit nur konstant viel Speicherplatz benötigt wird.

Aufgabe 5:
Gegeben sind ein Baum $T = (V, E)$, wobei V die Menge der Knoten und E die Menge der Kanten bezeichnet, sowie eine Menge P von einfachen Pfaden in T. Jeder Pfad $p_i \in P$ ist durch einen Startknoten $s_i \in V$ und einen Zielknoten $t_i \in V$, $t_i \neq s_i$ beschrieben und hat ferner ein Gewicht w_i. Gesucht ist eine Teilmenge S der Pfade, welche die Summe der Gewichte der Pfade maximiert, so dass keine zwei Pfade in S sich in einem Knoten oder einer Kante überschneiden. Formal suchen wir also eine Menge $S \subseteq P$, welche $\sum_{i|p_i \in S} w_i$ maximiert, unter der Bedingung, dass für alle $p_i, p_j \in S$, $p_i \cap p_j = \emptyset$.
Im Folgenden Beispiel sind fünf Pfade (*a-b*, *a-d*, *c-e*, *c-g*, *e-f*) gegeben. Die gesuchte Teilmenge S besteht aus den Pfaden *a-b* mit Gewicht 1, *c-g* mit Gewicht 10, und *e-f* mit Gewicht 3. Die Summe der Gewichte ist somit $\sum_{i|p_i \in S} w_i = 14$.



Hinweis: Wählen Sie einen Knoten r als Wurzel des Baums, und weisen Sie jedem Knoten $v \in V$ eine Höhe $h(v) \geq 0$ zu, welche gleich der Anzahl von Kanten auf dem Weg von v zu der Wurzel r in T ist. Also gilt $h(r) = 0$ und $h(v) > 0$ für alle $v \neq r$.

4 P a) Beschreiben Sie kurz in Worten einen Ansatz für einen möglichst effizienten Algorithmus nach dem Muster der dynamischen Programmierung, der, gegeben einen Baum T und eine Menge P von gewichteten, einfachen Pfaden in T, eine Menge $S \subseteq P$ von Pfaden berechnet, welche $\sum_{i|p_i \in S} w_i$ maximiert. Geben Sie die Rekursionsgleichung des dynamischen Programms an.

4 P b) Schreiben Sie nun einen dem in a) entwickelten Ansatz entsprechenden Algorithmus in Pseudocode auf. Die Eingabe besteht, wie oben beschrieben, aus einem Baum T und gewichteten Pfaden P. Der Algorithmus soll sowohl die oben beschriebene Teilmenge S von Pfaden als auch die Summe der Gewichte $\sum_{i|p_i \in S} w_i$ ausgeben.

1 P c) Geben Sie die Laufzeit der in b) implementierten Funktion an.