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

Musterlösung
Datenstrukturen und Algorithmen

9. Februar 2011

verfasst von Holger Flier (allfällige Korrekturen bitte an hflier@inf.ethz.ch)

Aufgabe 1:

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.

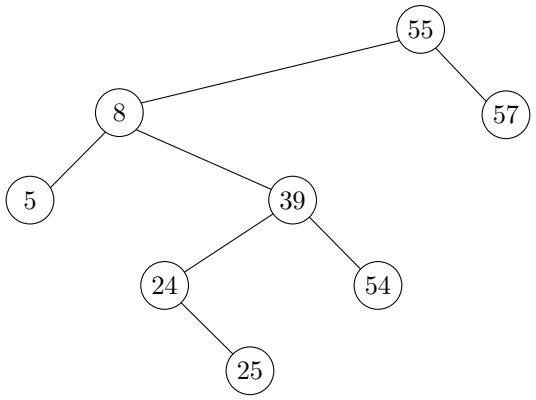
52	22	39	5	10	86	6	53	17	74	63	72	38

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

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: 14

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

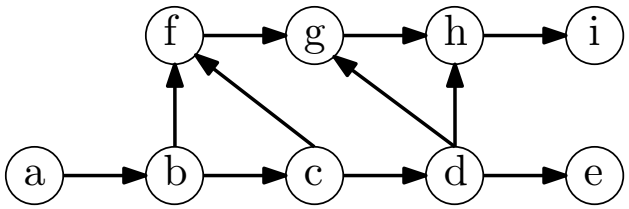


1 P d) Das untenstehende Array enthält die Elemente eines Min-Heaps in der üblichen Form gespeichert. Wie sieht das Array aus, nachdem das Minimum entfernt wurde und die Heap-Bedingung wieder hergestellt wurde?

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

13	15	64	52	38	70	73	78

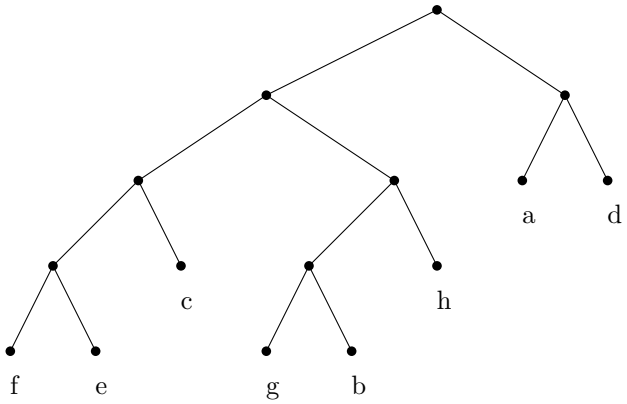
1 P e) Der folgende gerichtete Graph wird mit Tiefensuche traversiert. Die Suche startet beim Knoten a. Geben Sie eine Reihenfolge an, in der die Knoten erreicht werden können.



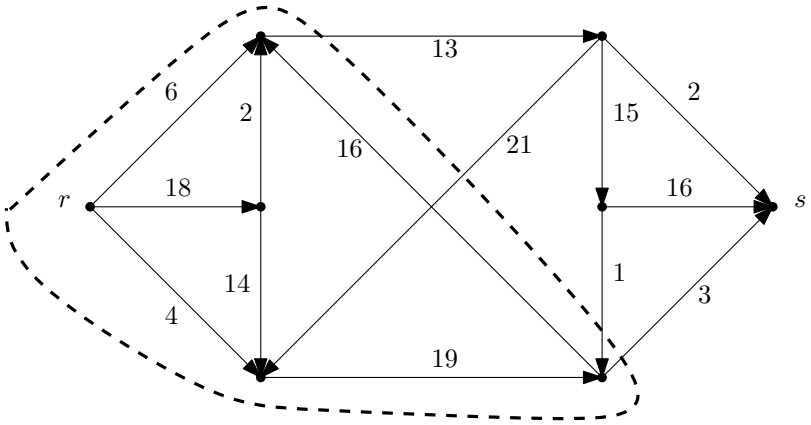
z.B.: a,b,f,g,h,i,c,d,e

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

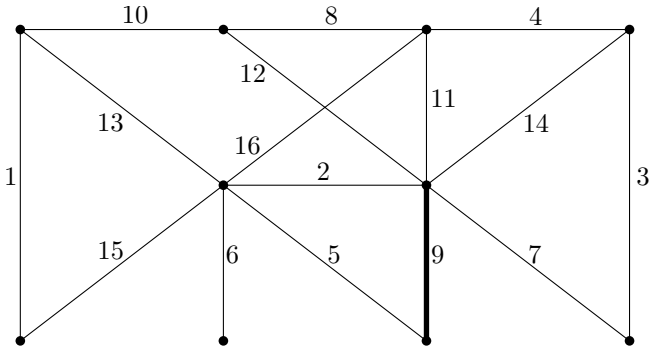


1 P g) Geben Sie den Wert eines maximalen (r, s)-Flusses in folgendem Netzwerk an und zeichnen Sie den entsprechenden minimalen Schnitt ein. Die Zahlen neben den Kanten geben die jeweilige Kantenkapazität an.

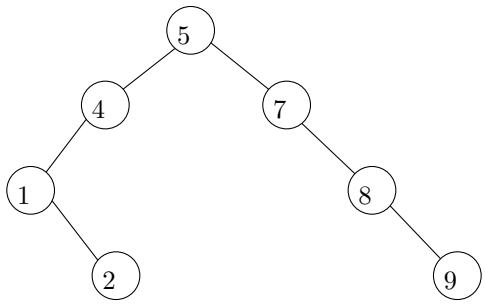


Der maximale Flusswert beträgt 16.

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)

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

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.

Lösung:

Durch Teleskopieren erhält man: $T(n) = \frac{9}{8}n^2 - \frac{n}{8}$

Beweis durch Induktion:

Induktionsverankerung:

$$T(1) = \frac{9-1}{8} = 1$$

Induktionsschritt:

$$T(n) = 4T(\frac{n}{2}) + \frac{n}{8} = 4(\frac{9n^2}{8 \cdot 4} - \frac{n}{2 \cdot 8}) + \frac{n}{8} = \frac{9n^2}{8} - \frac{2n}{8} + \frac{n}{8} = \frac{9n^2}{8} - \frac{n}{8}$$

1 P c) $\Theta(n \log n)$

1 P d) $\Theta(n^3)$

1 P e) $\Theta(n^{\frac{3}{2}})$

Aufgabe 3:
Hier bietet sich ein Greedy-Verfahren an, bei dem die Vorstellungen nach ihrer Endzeit (nicht-absteigend) geordnet werden. Sei π diese Ordnung, d.h. die o.g. Reihenfolge ist $v_{\pi(1)}, v_{\pi(2)}, \ldots, v_{\pi(n)}$. Wähle $v_{\pi(1)}$, und setze $e := e_{\pi(1)}$. Gehe für $i = 2, \ldots, n$ wie folgt vor. Falls $s_{\pi(i)} >= e$, so wähle $v_{\pi(i)}$ und setze $e := e_{\pi(i)}$.
Es ist leicht zu sehen, dass die so getroffene Auswahl optimal ist. Das Verfahren hat eine Laufzeit von $O(n \log n)$.

Aufgabe 4:

4 P a) Die Kundenbeziehungen lassen sich durch einen (gerichteten) Baum darstellen. Die Wurzel des Baums repräsentiert dabei die Bank, jeder andere Knoten einen Kunden. Eine Kante von Knoten u zu Knoten v im Baum bedeutet, dass Kunde v durch u geworben wurde.

Wir speichern den Baum in einem Array der Länge n ab. Im i-ten Eintrag des Arrays sind dabei, für alle $i = 1, \ldots, n$, zwei ganzzahlige Zähler (i^{in}, i^{out}) sowie ein Zeiger auf eine Liste gespeichert. Diese Liste speichert die Indizes der durch i geworbenen Kunden, also die Kinder von i im Baum. Die Zähler nennen wir Eingangs- und Ausgangszähler. Die Initialisierung des Arrays und der Listen benötigt $O(|S|) = O(|K|)$ Zeit. Die Initialisierung der Zähler geschieht durch eine Tiefensuche in dem Baum, startend bei der Wurzel. Bei dieser Tiefensuche wird ein globaler Zähler x immer dann um 1 erhöht, sobald ein neuer Knoten besucht wird oder ein bekannter Knoten abgearbeitet wurde. Wird ein Knoten zum ersten Mal besucht, so setze dessen Eingangszähler auf x. Ist ein Knoten abgearbeitet, so setze dessen Ausgangszähler auf x. Die Tiefensuche erfolgt in Linearzeit, d.h. die Initialisierung benötigt insgesamt $O(|K|)$.

Die Operation `nachfolger(`a`,`b`)` kann in $O(1)$ wie folgt realisiert werden. Sie gibt `true` zurück, falls $b^{in} < a^{in} < b^{out}$, sonst `false`.

2 P b) Hier genügt es, den entsprechenden Eintrag eines Kunden im Array als gelöscht zu markieren, sofern der Kunde keine Nachfolger hat. Die Operation `loesche(`u`)` läuft in $O(1)$. Eine Änderung an der Datenstruktur ist nicht nötig.

3 P c) Hier ändern wir die Datenstruktur wie folgt: Die Zähler sind nun rationale Zahlen beliebiger Genauigkeit. Die Operation `geworben(`a`,`b`)` bewirkt nun folgendes. Ein neuer Kunde $a \notin K$ wird in die Datenstruktur eingefügt. Sei $x := \max\{b^{in}, \max_{c \in C(b)} c^{out}\}$, wobei $C(b)$ die Menge der von b geworbenen Kunden beschreibt. Es gilt also $x < b^{out}$. Nun setzen wir die Zähler von a auf zwei beliebige Werte zwischen x und b^{out}, z.B. $a^{in} = x + \frac{1}{3}(b^{out} - x)$ und $a^{out} = x + \frac{2}{3}(b^{out} - x)$.

Das setzen der Zähler benötigt konstant viel Zeit, wenn c und b gegeben sind. Hat man b, und speichert man den zuletzt (vor a) geworbenen Kunden c am Anfang der Liste der von b geworbenen Kunden, so ist x in konstanter Zeit berechenbar. Es bleibt also, eine dynamische Datenstruktur zum Speichern der Kundeneinträge auszuwählen.

Ein balancierter Suchbaum benötigt $O(\log n)$ Zeit für Zugriff, Hinzufügen und Löschen. Somit ist eine Initialisierung in $O(|K| \log |K|)$ möglich, und alle drei Operationen können in $O(\log n)$ ausgeführt werden.

Aufgabe 5:

4 P a) Das dynamische Programm ist eine Induktion über alle Knoten, und zwar in nichtabsteigender Reihenfolge ihrer Höhe $h(v)$. Wir rechnen für jeden Knoten v den Wert $T[v]$ einer Optimallösung für das auf den Teilbaum T_v mit Wurzel v beschränkte Problem aus. Ferner rechnen wir für jeden Pfad p_i und jeden Knoten v den Wert $T[v,i]$ einer Optimallösung in T_v aus für den Fall, dass Pfad i gewählt ist.

Dabei wird für jeden Teilbaum T_v die Menge $P(v)$ von Pfaden bestimmt, welche v als höchsten Knoten enthalten (d.h., $h(v)$ ist am kleinsten). Die Rekursionsgleichung lautet wie folgt, wobei $C(v)$ die Menge der Kinder eines Knotens v beschreibt:

$$T[v,i] = \begin{cases} w_i + \sum_{c \in C(v) \cap p_i} T[c,i] + \sum_{c \in C(v) \setminus p_i} T[c], & \text{wenn } p_i \in P(v) \\ \sum_{c \in C(v) \cap p_i} T[c,i] + \sum_{c \in C(v) \setminus p_i} T[c], & \text{sonst} \end{cases}$$

wobei

$$T[v] = \max\{0, \sum_{c \in C(v)} T[c], \max_{i : p_i \in P(v)} T[v,i]\}$$

gilt. Die Summe der Gewichte der Pfade einer Optimallösung stehen in $T[r]$. Die ausgewählten Pfade lassen sich von $T[r]$ aus zurückverfolgen.

4 P b) Der Pseudocode ergibt sich aus der Beschreibung in a) und den folgenden Details. Sei n die Anzahl von Knoten und k die Anzahl von Pfaden. Die Funktion $h(v)$ lässt sich mittels Breitensuche in $O(n)$ berechnen. Damit berechnet man die Mengen $P(v)$ in $O(kn)$ — man startet für jeden Pfad p_i bei den Endknoten und geht im Baum hoch, bis man am höchsten Knoten des Pfades angelangt ist. Dabei legen wir auch eine Hilfstabelle an, in Zeit $O(nk)$, die für jeden Knoten v und jeden Pfad p_i speichert, ob $v \notin p_i$, $v \in p_i$, oder eben $p_i \in P(v)$.

1 P c) Gegeben n Knoten und k Pfade, so sind insgesamt $O(kn)$ Tabelleneinträge zu berechnen. Die Zeit zur Berechnung eines Tabelleneintrags $T[v,i]$ liegt in $O(n)$, in beiden Fällen. Zur Berechnung eines Eintrags in $T[v]$ benötigt man $O(k)$ Zeit. Insgesamt benötigt man also naiv $O(kn^2)$ Zeit. Da aber jedes Kind eines Knotens insgesamt in nur k Summen bei der Berechnung aller $T[v,i]$ vorkommt, benötigt man mit der o.g. Hilfstabelle in der Tat nur $O(kn)$ Zeit.