Technische Universität München
Institut für Informatik
Lehrstuhl für Effiziente Algorithmen

WS 2002/2003
Lösungen der Klausur
14. April 2003

Grundlegende Algorithmen

Aufgabe 1 (10 Punkte)

Betrachten Sie die Funktionen

$$f(n) = \begin{cases} 2f\left(\frac{n}{3}\right) + 3n \text{ für } n > 1 \\ 2 \text{ für } n = 1 \end{cases}$$

$$g(n) = \begin{cases} n^2 \text{ für } n \text{ gerade} \\ 2^n \text{ für } n \text{ ungerade} \end{cases}$$

Beweisen oder widerlegen Sie:

(a) $f(n) = O(n)$.

(b) $n^4 = O(g)$.

(c) $g = \omega(n \log n)$.

(d) $2^{3n} = o(3^{2n})$.

Lösungsvorschlag

(a) Wir zeigen $f(n) \leq 9n$ durch Induktion. Für $n = 1$ gilt $2 \leq 9$. Nun gilt $f(n) \leq 9n$ für $n < n_0$. Für $n = n_0$ erhalten wir

$$f(n) = 2f(\frac{n}{3}) + 3n \leq 2 \cdot 9\frac{n}{3} + 3n = 9n.$$

(b) Falls $n^4 \leq c \cdot g(n)$ für $n > n_0$ ist, dann gilt $16k^4 = O(4k^2)$, ein Widerspruch.

(c) Für die Funktion $h(n) = n^2$ gilt $h(n) \leq g(n)$ für alle n. Aus $\lim_{n\to\infty} \frac{n \log n}{n^2} = 0$ folgt $n \log n = o(h)$ und somit $g = \omega(n \log n)$.

(d) Es gilt $2^{3n} = 8^n$ und $3^{2n} = 9^n$. Somit gilt $\lim_{n\to\infty} \frac{2^{3n}}{3^{2n}} = \lim_{n\to\infty} \left(\frac{8}{9}\right)^n = 0$, also $2^{3n} = o(3^{2n})$.

Aufgabe 2 (10 Punkte)

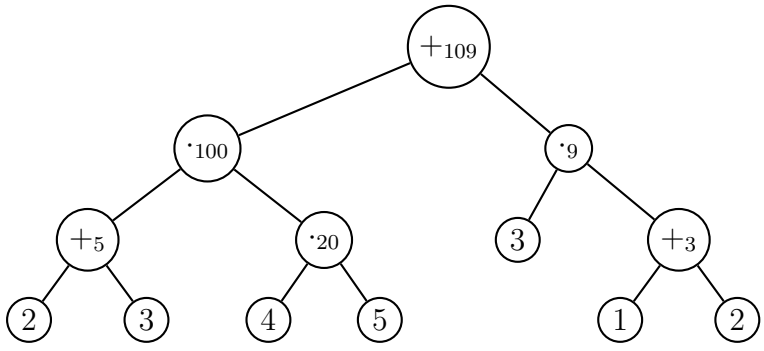
Ein *arithmetischer Baum* ist ein Binärbaum, bei dem jeder innere Knoten genau 2 Nachfolger hat und eines der Symbole $\cdot$ oder $+$ enthält. Jedes Blatt enthält eine ganze Zahl. Der Wert eines arithmetischen Baumes T ist induktiv wie folgt definiert: Falls T ein Blatt ist, das die Zahl z enthält, dann ist $W(T) = z$. Ansonsten seien T_1 und T_2 die beiden Teilbäume von T und $\circ \in \{\cdot, +\}$ das in der Wurzel von T gespeicherte Symbol. Der Wert von T ist $W(T) = W(T_1) \circ W(T_2)$.
Ein *arithmetischer Ausdruck* ist wie folgt definiert: Jede Zahl ist ein arithemtischer Ausdruck. Falls A und B arithmetische Ausdrücke sind, dann sind $(A + B)$ und $A \cdot B$ arithmetische Ausdrücke. Der Wert eines arithmetischen Ausdrucks ergibt sich durch Ausrechnen, wobei die üblichen Rechenregeln verwendet werden.
Ein arithmetischer Baum T ist genau dann zu einem arithmetischen Ausdruck A äquivalent, wenn $W(T) = W(A)$ gilt.

(a) Geben Sie zwei verschiedene arithmetische Bäume an, die zu dem arithmetischen Ausdruck $((2+3)\cdot 4\cdot 5+3\cdot(1+2))$ äquivalent sind. Wenden Sie die Funktion W auf beide Bäume an und zeichnen Sie an jedem Knoten v den von W gelieferten Wert für den mit v gewurzelten Teilbaum ein.

(b) Geben Sie einen Algorithmus an, der zu jedem arithmetischen Ausdruck einen äquivalenten arithmetischen Baum konstruiert.

Lösungsvorschlag

(a) Anwendung des Algorithmus aus Aufgabe (b) liefert den Baum



wobei die Zahlen in den Knoten jeweils den Wert darstellen, den W an diesem Knoten einnimmt. Ein zweiter Baum ist durch das Blatt mit Inhalt 109 gegeben.

(b) Eingabe ist eine Zeichenkette *expr*. Datenstruktur
ExpToTree($expr[], n$)
Falls die Länge von *expr* gleich 1 ist, dann wird das Blatt mit Eintrag $expr[0]$ zurückgegeben. Ansonsten Fallunterscheidung:

1. Fall $expr[0] =$'(': Der Ausdruck ist von der Form $(A + B)$ wobei A und B arithmetische Ausdrücke sind. Es wird die Position p von dem mittleren + bestimmt (siehe unten). Es wird ein neuer Knoten T angelegt, der das Symbol + enthält sowie ExprToTree($expr[0..p-1]$) als linken und ExprToTree($expr[p+1..n-2]$) als rechten Teilbaum.
2. Fall $expr[0] \neq$'(': Der Ausdruck ist von der Form $A \cdot B$ wobei A und B arithmetische Ausdrücke sind. Es wird die Position p des ersten, nicht in Klammern enthaltenen Symbol $\cdot$ bestimmt (siehe unten). Es wird ein neuer Knoten T angelegt, der das Symbol $\cdot$ enthält sowie ExprToTree($expr[0..p-1]$) als linken und ExprToTree($expr[p+1..n-2]$) als rechten Teilbaum.

Die Bestimmung der Positionen der Symbole + (1. Fall) und $\cdot$ (2. Fall) erfolgt, indem geklammerte Ausdücke überlesen werden, was schematisch wie folgt funktioniert. Die Variable i ist der Index des nächsten zu lesenden Zeichens, die Variable *count* zählt die Schachtelungstiefe und ist, nachdem die erste Klammer '(' gelesen wurde, auf 1 gesetzt:

```
(1)   while count > 0 do
(2)     if expr[i] ='(' then count := count + 1;
(3)     if expr[i] =')' then count := count - 1;
(4)     i := i + 1;
(4)   end
```

Aufgabe 3 (10 Punkte)

Zeigen Sie: Minimum und Maximum einer n-elementigen Menge können gleichzeitig mit maximal $n + \lceil \frac{n}{2} \rceil - 2$ Vergleichen bestimmt werden.

Lösungsvorschlag

Die Menge wird in Paare von 2 Elementen aufgeteilt. Fall n ungerade ist, wird die letzte Zahl z bei der Paarbildung nicht berücksichtigt.Nun wird das Maximum von jedem Paar bestimmt und die Menge der Maxima M_1 bzw. Minima M_2 gebildet (n ungerade: z wird zu beiden Mengen hinzugefügt). Nun wird das Maximum aus M_1 und das Minimum aus M_2 bestimmt. Falls n gerade ist, werden $\frac{n}{2} + \frac{n}{2} - 1 + \frac{n}{2} - 1 = \frac{3}{2}n - 2$ Vergleiche benötigt. Falls n ungerade ist, werden $\frac{n-1}{2} + \frac{n+1}{2} - 1 + \frac{n+1}{2} - 1 = n + \lceil \frac{n}{2} \rceil - 2$ viele Vergleiche benötigt.

Aufgabe 4 (10 Punkte)

Gegeben seien die Hashfunktion $h(k) = k \mod 11$ und die Folge $L = (1, 12, 7, 20, 2, 15, 24, 37, 40)$.

(a) Fügen Sie die Elemente von L der Reihe nach in die Hashtabelle H ein, wobei sie Kollisionen durch Verkettung auflösen. Wie viele Vergleiche sind im Worst-Case nötig, um ein Element zu finden.

(b) Fügen Sie die Elemente von L der Reihe nach in die Hashtabelle H ein, wobei sie Kollisionen durch lineare Sondierung auflösen. Wie viele Vergleiche sind im Worst-Case nötig, um ein Element zu finden.

Lösungsvorschlag

(a) Wir erhalten folgende Tabelle:

0	1	2	3	4	5	6	7	8	9	10
	[12,1]	[24,2]		[37,15]			[40,7]		20	

Es sind maximal 2 Vergleiche notwendig.

(b) Wir definieren $\bar{h}(k,i) = h(k) + i \mod 11 = k + i \mod 11$ (i ist die Anzahl der Versuche). Wir erhalten folgende Tabelle:

0	1	2	3	4	5	6	7	8	9	10
	1	12	2	15	24	37	7	40	20	

Es sind maximal 4 Vergleiche notwendig (bei $k = 24$).

Aufgabe 5 (10 Punkte)

Ist folgender Sortieralgorithmus korrekt? *Hinweis:* Achten Sie auf die Anzahl der Vergleiche.

```
(0)  BinSort(A[], l, r)
(1)  begin
(2)     n := (r − l) + 1;
(3)     if n < 3 then return MergeSort(A[]);
(4)     for i := 0 to (2⌈n/3⌉ − 1) do B[l + i] := A[l + i];
(5)     for i := 2⌈n/3⌉ to (n − 1) do C[l + i] := A[l + i];
(6)     Teile B wie bei QuickSort auf, wobei p der Rang des Pivotelements ist
(7)     BinSort(B[], 0, p − 1);
(8)     BinSort(B[], p, 2⌈n/3⌉ − 1);
(9)     return BinInsert(B[], C[]);
(10) end
```

Der Algorithmus BinInsert$(X[], Y[])$ fügt ein unsortiertes Feld Y der Größe n in ein sortiertes Feld X der Größe m ein, so dass ein sortiertes Feld entsteht, und benötigt dafür höchstens $m + 2n$ Vergleiche.

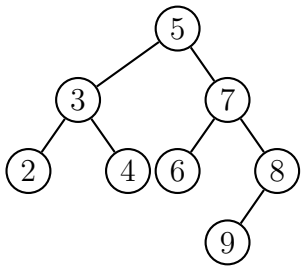
Lösungsvorschlag

Die Anzahl der Vergleiche im bestem Fall ergibt sich zu $V(n) = 2V(\frac{n}{3}) + \frac{2}{3}n - 1 + \frac{2}{3}n + \frac{2}{3}n \leq 2V(\frac{n}{3}) + 3n$. Nach Aufgabe 1(a) gilt $V(n) = O(n)$, ein Widerspruch zur unteren Schranke beim vergleichsbasiereten Sortieren.

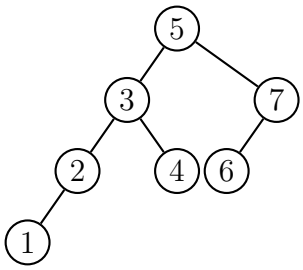
Aufgabe 6 (10 Punkte)

(a) Welche der folgenden Bäume sind AVL-Bäume? Begründen Sie Ihre Antwort.

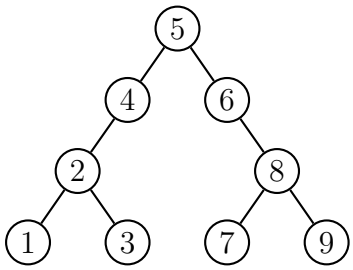
(i)



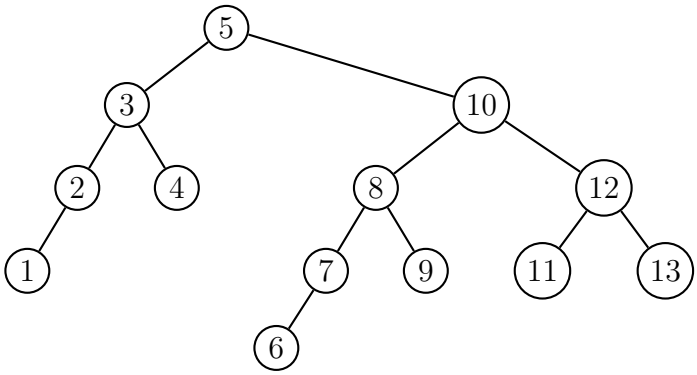
(ii)



(iii)



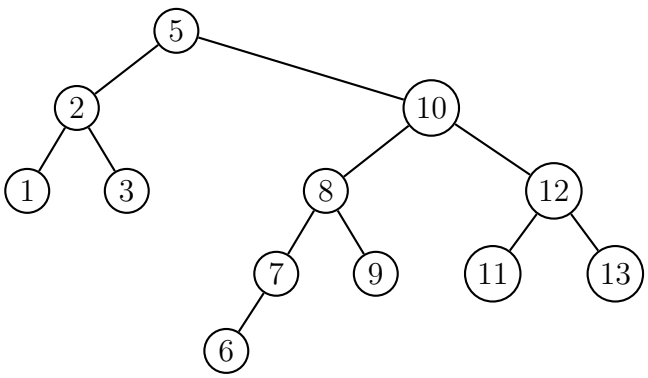
(b) Löschen Sie den Knoten mit Schlüssel 4 aus folgendem AVL-Baum und rebalancieren Sie, um wieder einen AVL-Baum zu erhalten. Dokumentieren Sie Ihre Vorgehensweise!



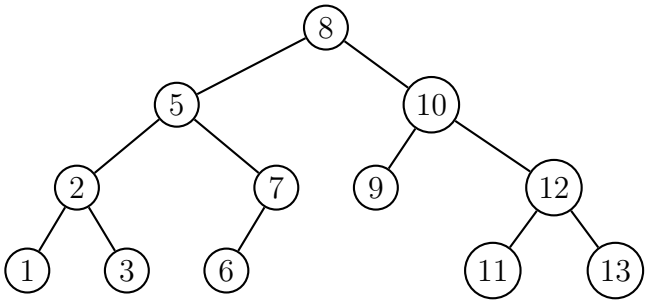
Lösungsvorschlag

(a) (i) ist kein AVL-Baum, da der Knoten 9 die Suchbaumeigenschaft verletzt. (ii) ist ein AVL-Baum, da die Suchbaumeigenschaft erfüllt ist und jeder Knoten höhenbalanciert ist. (iii) ist kein AVL-Baum, da die Knoten 4 bzw. 6 nicht höhenbalanciert sind.

(b) Nach dem Löschen des Knoten 4 ist der Knoten 3 nicht mehr höhenbalanciert, was durch einen Einfachrotation behoben werden kann. Es ergibt sich folgender Baum:



Nun ist der Knoten 5 nicht mehr höhenbalanciert was durch eine Doppelrotation korrigiert werden kann.



Aufgabe 7 (10 Punkte)

Gegeben sei der folgende Text T:

EINMAL_TRIFFT_EINSTEIN_WITTGENSTEIN

(a) Bestimmen Sie das kleinstmögliche dem Text T zugrunde liegende Alphabet Σ, sowie die Häufigkeit $h(a)$ für jeden Buchstaben $a \in \Sigma$.

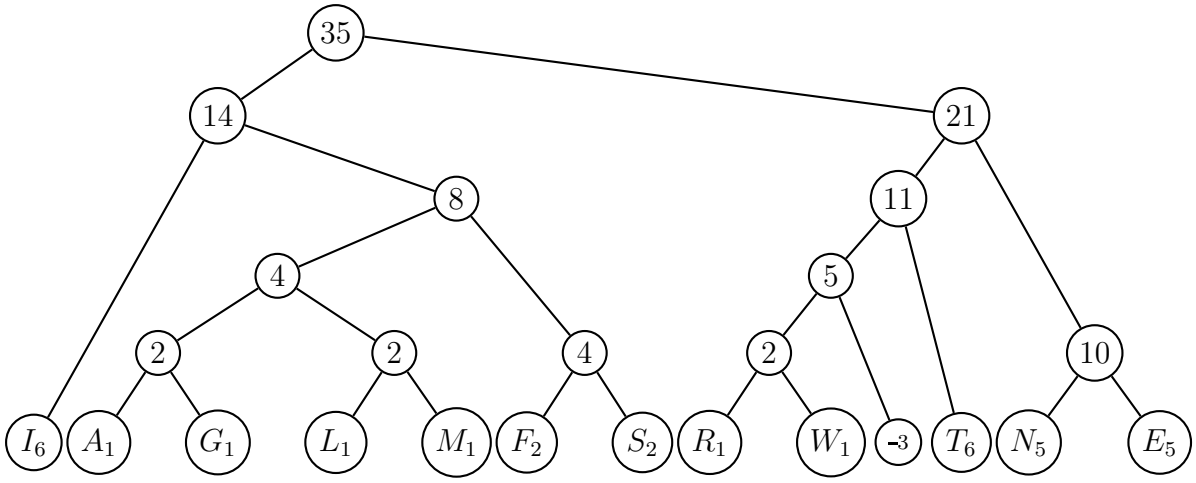
(b) Finden Sie einen optimalen Präfix-Code φ bzgl. T (bzw. h) über dem Alphabet $\{0, 1\}$ und berechnen Sie seine Länge $\sum_{a \in \Sigma} h(a)|\varphi(a)|$ (in Bits).

Lösungsvorschlag

(a) Das Alphabet ist $\Sigma = \{A, E, F, G, I, L, M, N, R, S, T, W, \_\}$. Die Häufigkeiten sind in folgender Tabelle aufgeführt.

Σ	A	E	F	G	I	L	M	N	R	S	T	W	_
h	1	5	2	1	6	1	1	5	1	2	6	1	3

(b) Mit Hilfe der Huffman-Codierung erhalten wir folgenden Baum:



Daraus ergibt sich folgender Code (links = 0, rechts = 1):

Σ	A	E	F	G	I	L	M
Code	01000	111	0110	01001	00	01010	01011
Σ	N	R	S	T	W	_	
Code	110	10000	0111	101	10001	1001	

Für die Länge gilt $\sum_{a\in\Sigma} h(a)|\varphi(a)| = 5{+}15{+}8{+}5{+}12{+}5{+}5{+}15{+}5{+}8{+}18{+}5{+}12 = 118$ Bits. Der Präfix-Code ist optimal, da das Huffman Verfahren benutzt wurde.

Aufgabe 8 (10 Punkte)

Betrachten Sie Wörter über dem einelementigen Alphabet $\Sigma = \{a\}$. Für das Wort $\underbrace{aa\ldots a}_{n\text{-mal}}$ der Länge n verwenden wir a^n als Abkürzung. Geben Sie in Abhängigkeit von n in Θ-Notation die Tiefe des Präfix-Baumes an, der bei Eingabe von a^n mit $n \geq 1$ vom Lempel-Ziv-Welch-Algorithmus konstruiert wird.

Lösungsvorschlag

Im i-ten Schritt hat der Baum die Höhe i, d.h., es werden genau i viele a's überlesen (falls die Eingabe genügend groß ist). Die Höhe h des Baumes ist daher

$$\sum_{i=1}^{m-1} i \leq h(n) \leq \sum_{i=1}^{m} i.$$

für ein passendes m. Da $\sum_{i=1}^{m} i = \frac{1}{2}(m-1)m = \Theta(m^2)$ ist, folgt $h(n) = \Theta\left(\sqrt{n}\right)$.