



Datenstrukturen und Algorithmen

Probeklausur im SS 2004
20. Juli 2004

Lehrstuhl für Informatik VI
der RWTH Aachen
Prof. Dr.-Ing. H. Ney

Hinweise zur Bearbeitung der Aufgaben:

- Diese Klausur ist eine *Auswahlklausur*. Von den insgesamt 112 Punkten werden 80 als volle Punktzahl angenommen.
- Wenn Sie aufgefordert werden einen *Algorithmus* zu *formulieren*, so tun Sie dies bitte in Java oder einer ähnlichen Sprache (C,C++). In jedem Fall dürfen Sie ausschließlich Konstrukte verwenden, die auch in Java zur Verfügung stehen. Verwenden Sie Kommentare um die Idee Ihres Algorithmus zu verdeutlichen.

Es kommt in erster Linie auf semantische Korrektheit und Verständlichkeit an. Syntaktische Feinheiten bis auf das letzte Semikolon sind nicht so wichtig. Konzentrieren Sie sich auf den Kern des Algorithmus; lassen Sie Ein- und AusgabeprozEDUREN weg.

- Verwenden Sie bei Bedarf das folgende

Master-Theorem

Sei $f(n)$ eine Funktion und $T(n)$ definiert durch die Rekursion

$$T(n) = aT(n/b) + f(n),$$

wobei $a \geq 1$ und $b > 1$ positive reelle Konstanten sind. Dann läßt sich $T(n)$ in asymptotischer Schreibweise wie folgt ausdrücken:

- Falls $f(n) \in O(n^{\log_b a - \varepsilon})$, $\varepsilon > 0$, dann ist $T(n) = \Theta(n^{\log_b a})$.
- Falls $f(n) \in \Theta(n^{\log_b a})$, dann ist $T(n) = \Theta(n^{\log_b a} \cdot \log_b n)$
- Falls $f(n) \in \Omega(n^{\log_b a + \varepsilon})$, $\varepsilon > 0$, und $af(n/b) \leq cf(n)$ mit $c < 1$ und genügend großem n , dann ist $T(n) = \Theta(f(n))$.

Aufgabe 1: Komplexität und Wachstumsklassen

5+4 Punkte

a) Beweisen Sie $(f, g, h : \mathbb{N} \rightarrow \mathbb{R}^+)$:

Transitivität: $g \in \Theta(h) \Rightarrow \Theta(g) \subseteq \Theta(h)$

Symmetrie: $f \in \Theta(g) \Leftrightarrow g \in \Theta(f)$

Umkehrsymmetrie: $f \in O(g) \Leftrightarrow g \in \Omega(f)$

b) Beweisen oder widerlegen Sie

$$\Theta(f) = \Omega(f) \cap O(f)$$

Aufgabe 2: Rekursionsgleichungen

6+4 Punkte

Schätzen Sie die Lösungen folgender Rekursionsgleichungen möglichst genau. Formulieren Sie Ihre Antwort mit Hilfe der Θ -Notation und begründen Sie sie.

a) $T(n) = 3T(n-2) + 2, \quad T(1) = T(2) = 2$

b) $T(n) = 7T\left(\frac{n}{3}\right) + n^2, \quad T(1) = 1$

Aufgabe 3: Wählen Sie ein MergeSort zusammen**6+10 Punkte**

Unten finden Sie die beiden Routinen `mergesort` und `merge`. Wählen Sie für beide Routinen in jeder Zeile eine der drei Möglichkeiten aus und schreiben Sie den entsprechenden Buchstaben in die letzte Spalte der Tabelle.

In jeder Zeile ist *genau* eine Möglichkeit richtig.

A	B	C	Antwort
<code>void mergesort(int [] a,int l, int r) {</code>			-
<code>if(l>r) {</code>	<code>if(l<r) {</code>	<code>if(l==r) {</code>	
<code>int m=(r+l)/2;</code>	<code>int m=(r-l)*2;</code>	<code>int m=r*l-2;</code>	
<code>mergesort(a,l,r);</code>	<code>mergesort(a,l,m);</code>	<code>mergesortT(a,r,l);</code>	
<code>mergesort(a,m-1,l);</code>	<code>mergesort(l,m,r);</code>	<code>mergesort(a,m+1,r);</code>	
<code>merge(a,l,m,r);</code>	<code>merge(a,l,r);</code>	<code>mergesort(a,l,m,r)</code>	
	<code>}}</code>	<code>}</code>	
<code>}</code>			-

A	B	C	Antwort
<code>void merge(int[] a,int l, int m, int r) {</code>			-
<code>char i,j;</code>	<code>int i,j;</code>	<code>double i,j;</code>	
<code>for(i=m+1;i>l;--i)</code> <code>{aux[i-1]=a[i-1];}</code>	<code>for(i=0;i<j;++i)</code> <code>{aux[i]=a[j];}</code>	<code>while(i<j)</code> <code>{ aux[++i]=a[--j];}</code>	
<code>for(m=j;m<r;++m)</code> <code>{aux[r-m+j]=a[m+1];}</code>	<code>for(j=r;r<j;++r)</code> <code>{aux[r-m+j]=a[2*j];}</code>	<code>for(j=m;j<r;++j)</code> <code>{aux[r+m-j]=a[j+1];}</code>	
<code>for(int k=0;k<l;++k) {</code>	<code>for(int k=0;k<r;++k) {</code>	<code>for(int k=l;k<=r;++k) {</code>	
<code>if(i<j) {</code>	<code>if(aux[j] < aux[i]) {</code>	<code>if(j<i) {</code>	
<code>a[k]=aux[j];</code>	<code>aux[k]=a[j];</code>	<code>a[k]=a[j];</code>	
<code>--j;</code>	<code>++j;</code>	<code>--m;</code>	
<code>} end;</code>	<code>} else if(j==i) {</code>	<code>} else {</code>	
<code>aux[k]=a[i];</code>	<code>a[k]=aux[i];</code>	<code>a[i]=aux[k];</code>	
<code>++j;</code>	<code>++i;</code>	<code>++aux;</code>	
<code>}}</code>			-

Aufgabe 4: Sortieren geht über Studieren**5+5 Punkte**

Sortieren Sie die Folge

85, 1, 7, 2, 83, 19, 14, 12

mit

- MergeSort. Geben Sie dabei nach jedem Aufruf von `merge` die Folge und die Werte für linke und rechte Grenze (l bzw. r) des Teilfeldes in Tabellenform an.
- SelectionSort. Geben Sie nach jeder Vertauschung von Elementen die Folge in Tabellenform an.

Aufgabe 5: Optimaler Suchbaum**10 Punkte**

Gesucht sei ein optimaler binärer Suchbaum mit vier Knoten sowie den Häufigkeiten der Knoten ($\beta_i, i = 1, \dots, 4$) bzw. der "Nieten" (erfolglose Zugriffsversuche) ($\alpha_i, i = 0, \dots, 4$) mit:

i	0	1	2	3	4
β_i		3	1	7	11
α_i	0	2	3	5	4

Berechnen Sie einen optimalen Suchbaum. Stellen Sie den Baum graphisch dar.

Aufgabe 6: Hashing

2+8 Punkte

- a) Beschreiben Sie den Unterschied zwischen offenem und geschlossenem Hashing.
 b) Fügen Sie die Zahlenfolge

1, 20, 35, 17, 9, 18, 25, 41, 50, 22, 21, 15, 49, 8, 28

in die auf dem Lösungsblatt gegebene Hashtabelle ein. Die Hashfunktion ist

$$h(k) = k \pmod{17}.$$

Als Sondierungsstrategie soll quadratisches, alternierendes Sondieren (dieses wurde in der Vorlesung auch als quadratisches Sondieren mit Verfeinerung eingeführt) verwendet werden. Geben Sie für jeden Schlüssel alle Hashwerte an, die berechnet werden, *bis ein freier Tabellenplatz gefunden wird*. Bestimmen Sie die Gesamtzahl der nötigen Sondierungsschritte.

Resultierende Hashtabelle:

Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Schlüssel k																	

Anzahl Sondierungsschritte: _____

Aufgabe 7: Effiziente Suche

8+2 Punkte

Die Inhaber einer elektronischen Buchhandlung haben in einem großen Feld A alle üblichen Buchpreise von 0 bis 1000 Euro aufsteigend sortiert ohne Wiederholungen eingetragen.

- a) Formulieren Sie einen sehr effizienten Algorithmus, der nach einem bestimmten Preis in A sucht. Ihr Algorithmus soll die Eigenschaften der gegebenen Daten ausnutzen. Erklären Sie kurz die Funktionsweise Ihres Algorithmus.
 b) Was ist die obere Schranke für die Zeitkomplexität Ihres Algorithmus im “average case” (ohne Beweis)?

Aufgabe 8: Breitendurchlauf

8 Punkte

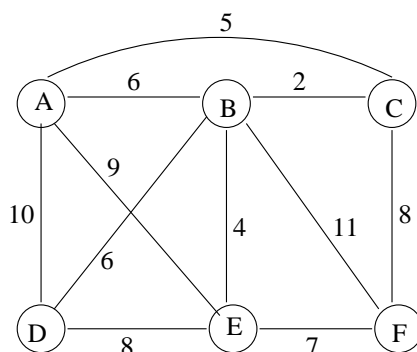
Geben Sie für die folgenden Aussagen an, ob sie wahr oder falsch sind:

Für eine richtige Antwort wird ein Punkt gezählt, für eine falsche wird ein Punkt abgezogen. Enthaltungen sind neutral. Die Summe der Punkte dieser Aufgabe kann nicht negativ werden.

Breitendurchlauf von Graphen...	wahr	falsch	Enthaltung
... wird mittels eines Stacks realisiert.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
... wird mittels einer Queue realisiert.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
... läuft genau so ab wie ein Tiefendurchlauf.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
... läßt sich auf Bäume anwenden.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
... erfordert das Markieren der schon besuchten Knoten.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
... besucht zunächst alle Nachfolger eines Knotens, dann deren Nachfolger usw.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
... ist für Graphen mit Zyklen nicht durchführbar.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
... kann nicht iterativ gelöst werden.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Aufgabe 9: Minimaler Spannbaum**6 Punkte**

Es sei der folgende gewichtete Graph $G = (V, E)$ gegeben.



Bestimmen Sie einen minimalen Spannbaum. Machen Sie die einzelnen Schritte des Algorithmus in Tabellenform deutlich.

Aufgabe 10: Editierabstand**8 Punkte**

Berechnen Sie den Editierabstand (minimale Anzahl von Ersetzungen, Löschungen und Einfügungen) zwischen den folgenden Buchstabenfolgen:

ADRESSE
REGISTER

Geben Sie außerdem alle notwendige Editieroperationen an.

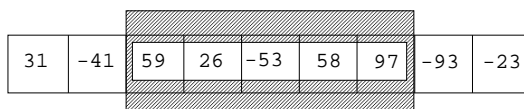
Aufgabe 11: Maximale Teilfolge**2+3+10 Punkte**

Gegeben sei eine Liste ganzer Zahlen a_k der Länge n ($k \in \{1, \dots, n\}$). Die Teilfolgensumme von a in den Grenzen $i \leq j$ ist folgendermaßen definiert:

$$S_a(i, j) := \sum_{k=i}^j a_k$$

Gesucht wird die Teilfolge a_i, \dots, a_j , deren Summe $S_a(i, j)$ maximal ist; wir nennen diese die *maximale Teilfolge*.

Beispiel:



Der schraffierte Bereich stellt für das gegebene Beispiel die maximale Teilfolge dar ($i = 3, j = 7$).

- Wieviele Teilfolgen existieren überhaupt?
- Geben Sie eine untere Schranke für die Komplexität eines Verfahrens zur Bestimmung der maximalen Teilfolge an.
- Entwerfen Sie einen Algorithmus zur Bestimmung der maximalen Teilfolge mit möglichst geringer Komplexität.