Institut für Informatik der Universität München
Prof. Martin Hofmann
Dr. Andreas Abel
Dipl.-Inf. Roland Axelsson

SS 2009

4. Juni 2009

Probeklausur zur Vorlesung
Algorithmen und Datenstrukturen

Schreiben Sie bitte auf jedes abgegebene Blatt **Namen** und **Übungsgruppe**! Heften Sie die Blätter vor Abgabe zusammen. Viel Erfolg!

Aufgabe 1: (8 Punkte) Bestimmen Sie mit Hilfe des Master-Theorems für die folgenden Rekursionsgleichungen möglichst scharfe asymptotische untere und obere Schranken, falls das Master-Theorem anwendbar ist. Geben Sie andernfalls eine kurze Begründung, warum das Master-Theorem nicht anwendbar ist.

a) $T(n) = 16T(n/2) + 40n - 6$

b) $T(n) = 27T(n/3) + 3n^3 \log n$

c) $T(n) = 4T(n/2) + 3n^2 + \log n$

d) $T(n) = 4T(n/16) + 100 \log n + \sqrt{2n} + n^{-2}$

Aufgabe 2: (8 Punkte) Gegeben seien die folgenden Funktionen:

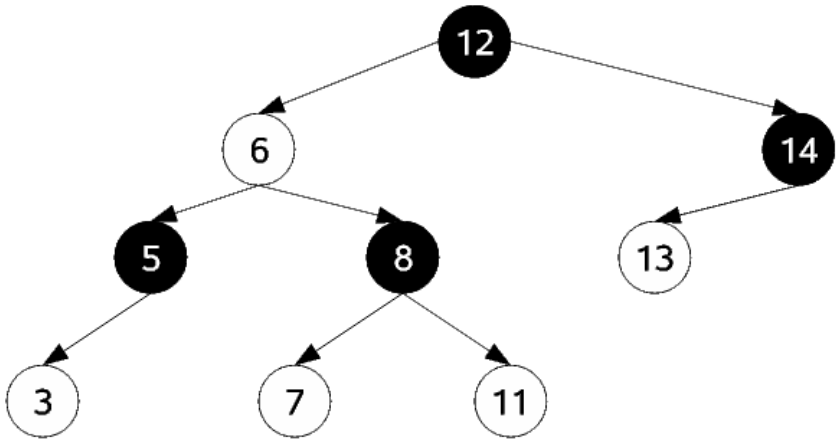
$$f_1 = n \log(n^4) \qquad f_2 = 4n^{1/2} \qquad f_3 = 1/2 \log(n!)$$
$$f_4 = 4n^2 \log n \qquad f_5 = 2^{(\log n)^2} \qquad f_6 = 1/2\, n^{31/15}$$

Welche der folgenden Aussagen gelten? Sie müssen ihre Antworten nicht begründen, es gibt jedoch für jede falsche Antwort einen Punkt Abzug. Insgesamt wird aber diese Aufgabe nicht mit weniger als 0 Punkten bewertet.

a) $f_5(n) = \Theta(f_3(n))$ b) $f_6(n) = O(f_4(n))$ c) $f_1(n) = O(f_3(n))$
d) $f_2(n) = \Omega(f_3(n))$ e) $f_4(n) = O(f_1(n))$ f) $f_5(n) = \Omega(f_6(n))$
g) $f_1(n) = \Omega(f_4(n))$ h) $f_2(n) = O(f_6(n))$

Aufgabe 3: (8 Punkte)

a) Zeichnen Sie den untenstehenden Rot-Schwarz-Baum ab und markieren Sie die Makroknoten (aus dem entsprechenden 2-3-4-Baum). [Die roten Knoten sind weiß gezeichnet.]



b) Zeichnen Sie die beiden Rot-Schwarz-Bäume, die entstehen, wenn man nacheinander die Schlüssel 2 und 1 in den obigen Baum gemäß dem Einfügealgorithmus für Rot-Schwarz-Bäume aus der Vorlesung einfügt.

c) Zeichnen Sie die beiden Rot-Schwarz-Bäume, die entstehen, wenn man nacheinander die Schlüssel 7 und 6 aus dem in Teilaufgabe a) angegebenen Rot-Schwarz-Baum gemäß dem Löschalgorithmus für Rot-Schwarz-Bäume aus der Vorlesung entfernt.

Hinweis: Falls Sie die Zwischenschritte geeignet dokumentieren, können auch für teilweise richtige Lösungen entsprechend Punkte vergeben werden.

Aufgabe 4: (12 Punkte) Ein Programmierer hat in die Implementierung von Merge-Sort eine einzige Programmzeile eingefügt, nämlich Zeile 5 im folgenden Codestück— die Funktion MERGE wurde nicht verändert, und ist daher nicht gezeigt:

```
MERGE-SORT-O(a, p, r)
1  if p < r
2    then q ← ⌊(p + r)/2⌋
3         MERGE-SORT-O(a, p, q);
4         MERGE-SORT-O(a, q + 1, r);
5         if a[q] > a[q + 1]
6           then MERGE(a, p, q, r);
```

a) Geben Sie je ein kurzes Argument an, das zeigt, dass kein Zugriff auf Indices außerhalb der Arraygrenzen erfolgt, und der Algorithmus weiterhin alle Eingaben korrekt sortiert.

b) Geben Sie für jedes n ein n-elementiges Array A_n an, für das der Algorithmus die bestmögliche Laufzeit hat, d.h. Zeile 6 wird niemals ausgeführt.

c) Sei $T(n)$ die spezielle Laufzeit für die Eingabe A_n. Stellen Sie eine Rekursionsgleichung für $T(n)$ auf, und bestimmen Sie mit Hilfe des Master-Theorems eine einfache Funktion f, so dass $T(n) = \Theta(f(n))$. Vergleichen Sie dies mit der best-case Laufzeit von Merge-Sort.

d) Wir möchten nun untersuchen, wie sinnvoll diese Optimierung für den durchschnittlichen Fall ist. Für ein Array mit den Elementen aus der Menge $\{1, 2, \ldots 2n\}$ müssen *ganz am Ende des Ablaufs* des Algorithmus zwei sortierte Arrays mit je n Elementen mittels MERGE verschmolzen werden, genau dann wenn $a[n] > a[n+1]$— andernfalls greift die Optimierung. Geben Sie für ein zufällig gewähltes Array mit paarweise verschiedenen Elementen aus $\{1, 2, \ldots 2n\}$ eine exakte Formel für die Wahrscheinlichkeit an, dass die Optimierung dieses eine Mal greift, der zusätzliche Vergleich also nicht überflüssig war. Zeigen Sie, dass diese Wahrscheinlichkeit bereits für für $n = 5$, also für ein zufällig gewähltes Array mit 10 Elementen, kleiner als 1% ist.

Aufgabe 5: (6 Punkte) Professor Geröllheimer hat sich das Konzept der quasi-balancierten Bäume überlegt:

Ein binärer Suchbaum ist quasi-balanciert, falls die Höhendifferenz zwischen linkem und rechtem Kind der Wurzel -1,0, oder 1 beträgt. Diese Eigenschaft muss *nur* für die Kinder der Wurzel gelten.

Wenn in einen quasi-balancierten binären Suchbaum ein Knoten in den Teilbaum links unter der Wurzel eingefügt wurde, wird dadurch eventuell die Quasi-Balanciertheit zerstört. Wenn dies eintritt, wird der Baum um die Wurzel nach rechts rotiert. Wenn in einen quasi-balancierten Baum ein Knoten in den Teilbaum rechts unter der Wurzel eingefügt wurde, und dadurch die Quasi-Balanciertheit zerstört wurde, wird der Baum um die Wurzel nach links rotiert.

Der Professor behauptet, dass damit für jeden binären Suchbaum, der vorher quasi-balanciert war, nach dem Einfügen die Quasi-Balanciertheit wiederhergestellt wird. Hat der Professor recht? Beweisen Sie die Aussage, oder geben Sie ein Gegenbeispiel an.

Aufgabe 6: (8 Punkte) Ein Weinkellner benutzt das folgende System zur Lagerung seiner Weine. Die $n \geq 8$ Flaschen werden in 3 Kategorien A, B und C eingeteilt:

> Kategorie A soll *möglichst wenige* Flaschen enthalten, die zusammen mindestens 60% des Wertes ausmachen — falls dies möglich ist: Diese werden dann in einem speziellen Weinkühlschrank gelagert, dessen Kapazität $n/\log n$ Flaschen fasst. Falls die $n/\log n$ teuersten Flaschen also zusammen einen Wert von weniger als 60% haben, fallen eben die $n/\log n$ teursten Flaschen in Kategorie A.
>
> Kategorie C enthält die 60% der Flaschen, d. h. $\lfloor 0.6n \rfloor$ *Stück, die den geringsten Wert haben.* Diese sind zum alltäglichen Genuss bestimmt und werden aufrecht stehend im Vorratsschrank gelagert.
>
> Die restlichen Flaschen bilden die Kategorie B, diese lagern im Keller im Weinregal.

Ein möglicher Algorithmus zur Klassifikation der Weinflaschen besteht darin, diese zuerst (möglichst effizient) *nach ihrem Wert zu sortieren.* Dann wird solange die jeweils wertvollste Flasche zur Kategorie A hinzugefügt, bis 60% des Wertes oder $n/\log n$ Flaschen erreicht sind. Anschließend wird $\lfloor 0.6n \rfloor$ mal die Flasche mit dem geringsten Wert in die Kategorie C eingereiht. Die verbleibenden Flaschen bilden Kategorie B.

a) Konstruieren Sie ein Beispiel mit $n = 8$ Flaschen, denen Sie Werte so zuordnen, dass mehr als 60% des Gesamtwertes in Kategorie A fallen.

b) Geben Sie einen Algorithmus zur Klassifikation an, der eine asymptotisch bessere Laufzeit im worst-case hat, und begründen Sie dies. Sie können aus der Vorlesung bekannte Algorithmen und Datenstrukturen verwenden.