INSTITUT FÜR INFORMATIK
KOMPLEXITÄT UND KRYPTOGRAFIE
PROF. DR. JOHANNES KÖBLER

SS 2008
28. MAI 2008

Theoretische Informatik 3

4. Übung

Abgabe der schriftlichen Lösungen bis zum 18. Juni 2008

Aufgabe 21 [*mündlich*]

Entscheiden Sie für folgende Sortierverfahren, ob sie stabil sind (jeweils mit Begründung):

a) InsertionSort, b) BubbleSort c) MergeSort, d) QuickSort,
e) HeapSort, f) CountingSort, g) RadixSort, h) BucketSort.

Aufgabe 22 [*mündlich*]

Bestimmen Sie für $n = 1, \ldots, 10$ die Fragekomplexität von HeapSort im besten, schlechtesten und durchschnittlichen Fall für eine Permutation der Zahlen $1, \ldots, n$. Vergleichen Sie diese Werte jeweils mit den entsprechenden Werten für QuickSort aus Aufgabe 17 d). Welche Werte ergeben sich für die Floyd-Strategie?

Aufgabe 23 [*mündlich*]

Bestimmen Sie asymptotische Schranken für die Anzahl von Vergleichen, die HeapSort im besten Fall vornimmt, um eine Folge von n paarweise verschiedenen Zahlen zu sortieren. Welche Schranke ergibt sich, wenn die Folgenglieder auch mehrfach vorkommen können?

Aufgabe 24 [*3 Punkte*]

Sei $k \geq 2$. Ein k-Heap H ist wie ein (binärer) Heap definiert, nur dass jeder Knoten (bis zu) k Kinder hat. Bestimmen Sie (jeweils mit Begründung):

a) Die zur Speicherung von H in einem Feld benötigten Indexfunktionen $\texttt{parent}(i)$ und $\texttt{child}(i, j)$, $j = 1, \ldots, k$, wobei $\texttt{child}(i, j)$ den Index des j-ten Kindes des Knotens mit Index i angibt. (*mündlich*)

b) Die Tiefe eines k-Heaps mit n Knoten. (*3 Punkte*)

Aufgabe 25 [*mündlich*]

a) Schätzen Sie die erwartete Laufzeit von BucketSort für den Fall ab, dass eine Folge von n im Intervall $[a, b)$ unabhängig gleichverteilten Zufallszahlen auf m Buckets verteilt wird und die Buckets mit einem Verfahren der Komplexität $O(n^2)$ sortiert werden.

b) Wie groß ist die erwartete Laufzeit bei Verwendung von $m = \sqrt{n}$ Buckets?

Aufgabe 26 [*mündlich*]

Implementieren Sie die Prozeduren `Max`, `Min`, `Prec`, `Succ` und `Delete` für verkettete Listen und binäre Suchbäume. Geben Sie jeweils asymptotische Schranken für die Laufzeit der Prozeduren in Abhängigkeit von der Listenlänge bzw. der Tiefe des Suchbaums an.

Aufgabe 27 Zeigen Sie: [*mündlich*]

a) Wird als Einfügesequenz eine zufällige Permutation von n verschiedenen Zahlen benutzt, so hat der resultierende binäre Suchbaum B eine mittlere Knotentiefe von $O(\log n)$.

Hinweis: Zeigen Sie, dass die mittlere Knotentiefe von B mit der durchschnittlichen Anzahl von Vergleichen von QuickSort beim Sortieren einer zufälligen Permutation von n verschiedenen Zahlen übereinstimmt.

b) Die mittlere Laufzeit der Prozedur $\texttt{Search}(B, k)$ ist $O(\log n)$, falls B aus einer zufälligen Einfügesequenz von n verschiedenen Zahlen generiert wurde und k ein zufälliger Wert dieser Folge ist.

Aufgabe 28 [*7 Punkte*]

Implementieren Sie eine Prioritätswarteschlange $P = (H, s)$ mit Hilfe eines in einem Feld H gespeicherten Heaps (s soll die aktuelle Größe von P angeben). P soll bis zu n Schlüsselwerte aufnehmen können. Geben Sie Pseudocode für folgende Prozeduren an.

a) Die Prozedur $\texttt{Insert}(H, s, k)$ fügt ein neues Element mit dem Wert k in den Heap ein und aktualisiert den Wert von s.

b) Die Funktion $\texttt{Max}(H)$ liefert den maximalen in P gespeicherten Schlüsselwert zurück.

c) Die Funktion $\texttt{DeleteMax}(H, s)$ liefert das Element mit dem maximalen in P gespeicherten Schlüsselwert zurück und entfernt dieses Element aus H. Zudem wird s aktualisiert und die Heap-Eigenschaft wieder hergestellt.

d) Die Prozedur $\texttt{IncreaseKey}(H, i, k)$ erhöht den Wert von $H[i]$ auf das Maximum von $H[i]$ und k und stellt die Heap-Eigenschaft wieder her.

e) Die Prozedur $\texttt{ChangeKey}(H, s, i, k)$ setzt den Wert von $H[i]$ auf k und stellt die Heap-Eigenschaft wieder her.

Geben Sie jeweils asymptotische Schranken für die Laufzeit der Prozeduren an.

Aufgabe 29 [*mündlich*]

Geben Sie einen möglichst effizienten Algorithmus an, der k sortierte Listen zu einer sortierten Liste zusammenfügt.

Hinweis: Verwenden Sie die Prioritätswarteschlange aus voriger Aufgabe. Falls n die Gesamtzahl aller Elemente in den k Listen bezeichnet, ist dies in Zeit $O(n \log k)$ möglich.