Datenstrukturen & Algorithmen, WS 1999/2000

Vorlesungsprüfung

6. Oktober 2000

Es sind keinerlei Unterlagen und Hilfsmittel zugelassen. Zeit: 90 min

Notenschlüssel: Genügend ab 11 Punkten, Befriedigend ab 13 Punkten, Gut ab 15 Punkten, Sehr gut ab 17 Punkten. [20 Punkte möglich]

1. Nehmen Sie an, daß die Elemente eines linearen Feldes $\texttt{A[1..}n\texttt{]}$ schon folgendermaßen teilsortiert sind:

   $$\texttt{A[1]} \geq \texttt{A[2]} \geq \cdots \geq \texttt{A[}k\texttt{]}$$
   $$\texttt{A[}k+1\texttt{]} \leq \texttt{A[}k+2\texttt{]} \leq \cdots \leq \texttt{A[}n\texttt{]}$$

   Des weiteren sei $\texttt{A[}k\texttt{]} \geq \texttt{A[}n\texttt{]}$ wobei k *unbekannt* ist.

   (a) Wie lange benötigt MERGESORT *ordnungsmäßig* zum Sortieren dieses Feldes, in Abhängigkeit von n und k? [1 Punkt]

   (b) Wie lange benötigt INSERTION-SORT *ordnungsmäßig* zum Sortieren dieses Feldes, in Abhängigkeit von n und k? [1 Punkt]

   (c) Geben Sie einen Algorithmus an, der das Feld in linearer Zeit sortiert. [2 Punkte]

   (d) Geben Sie einen Algorithmus an, der in konstanter Zeit den Median des Feldes berechnet. [1 Punkt]

2. Ein lineares Feld $\texttt{A[1..}n\texttt{]}$ enthält eine beliebige Folge der Buchstaben `a`, `b`, und `c`.

   Gesucht ist eine Aufteilung des Feldes in 3 Abschnitte

   $\texttt{A[1..}n_1\texttt{]}$, $\texttt{A[}n_1+1\texttt{..}n_2\texttt{]}$, $\texttt{A[}n_2+1\texttt{..}n\texttt{]}$,

   sodaß folgende Summe maximal ist:

   (Anzahl der `a`'s in $\texttt{A[1..}n_1\texttt{]}$)
   + (Anzahl der `b`'s in $\texttt{A[}n_1+1\texttt{..}n_2\texttt{]}$)
   + (Anzahl der `c`'s in $\texttt{A[}n_2+1\texttt{..}n\texttt{]}$)

   Geben Sie einen möglichst schnellen Algorithmus zur Berechnung einer optimalen Aufteilung an. Wieviel Zeit benötigt Ihr Algorithmus im ungünstigsten Fall? Begründung!

   [$O(n^3)$: 1 Punkt — $O(n^2)$: 3 Punkte — $O(n)$: 5 Punkte]

3. Nehmen Sie an, Sie verwenden eine ideale Hash-Funktion h um n verschiedene Elemente in einer Hash-Tabelle der Größe n mit Überläuferlisten zu speichern (d.h. $\text{Prob}\{h(w) = j\} = 1/n$ für alle $j = 1, \ldots, n$).

   a) Sei f_i die Anzahl der freien Plätze in der Hash-Tabelle nach dem Einfügen von i Elementen. Wie groß ist die Wahrscheinlichkeit, dass nach dem Einfügen des $(i+1)$-ten Elementes die Anzahl der freien Plätze sinkt? [1 Punkt]

   b) Sei F_i die *erwartete* Anzahl von freien Plätze in der Hash-Tabelle nach dem Einfügen von i Elementen. Verwenden Sie das Ergebnis aus a) um eine Rekursionsgleichung für F_i aufzustellen. [2 Punkte]

   c) Berechnen Sie F_n explizit. [1 Punkt]

   d) Verwenden Sie die Ergebnisse aus a)–c) und geben Sie an, bei ca. wieviel Prozent der n eingefügten Elemente es zu einer Kollision kommt? Nehmen Sie an, dass n sehr groß ist. [1 Punkt]

4. Zwei Fastfood-Ketten (M und B) haben sich geeinigt in einem noch unberührten Land ihre Restaurants folgendermaßen zu errichten:

   - in jedem Ort soll es nur ein Restaurant geben
   - sind zwei Orte benachbart, so sollen die Restaurants in diesen Orten zu verschiedenen Fastfood-Ketten gehören.

   Sie sollen nun einen Algorithmus (möglichst detailierter Pseudocode) entwerfen, der überprüft, ob eine solche Aufteilung möglich ist. Zu jedem Ort `x` des Landes gibt es eine Liste `x->n[1..k]` der `k` Nachbarorte. Sie können beliebige Zusatzinformationen in Zusammenhang mit einem Ort speichern. Z.B. könnte `x->r` angeben zu welcher Fastfood-Kette (M oder B) das Restaurant im Ort `x` schlußendlich gehört. Weiters dürfen Sie die drei Funktionen `push(S,x)`, `pop(S)` und `isempty(S)` für einen beliebigen Stapel `S` verwenden. [5 Punkte]