Informatik II SS 2006

Übungsblatt 8

Abgabe: 27.6.06 - 30.6.06

Aufgabe 1 (Untere Schranke für vergleichsbasierte Sortierverfahren) Punkte: 2+3

a) Geben Sie die geringstmögliche Tiefe eines Blattes in einem Entscheidungsbaum für vergleichsbasierte Sortierverfahren abhängig von der Eingabelänge n an. Begründen Sie ausführlich. Interpretieren Sie die minimale Tiefe hinsichtlich des Aufwands von Sortierverfahren.

b) Zeichnen die den Entscheidungsbaum für Quicksort und Mergesort bzgl. einer 3-elementigen Eingabe auf. Wählen Sie bei Quicksort immer das Element mit dem größten Index einer Partition als Pivot-Element. Markieren Sie in beiden Entscheidungsbäumen den zur Eingabe (1,2,0) gehörenden Pfad.

Aufgabe 2 (Counting-Sort) Punkte: 1+2+2

Gehen Sie im folgenden von der Countingsort-Implementierung aus der Vorlesung aus

a) Sortieren Sie die Eingabefolge A = (5,1,3,1,2,0,6,5,2,0,3) mit Counting-Sort. Geben Sie das Hilfsarray C nach Ausführung der ersten Schleife ($C[i] = |\{a_j \in A | a_j = i\}|$) und der zweiten Schleife ($C[i] = |\{a_j \in A | a_j \leq i\}|$) an.

b) Ist Counting-Sort ein stabiles Sortierverfahren? Begründen Sie Ihre Antwort.

c) Angenommen der Kopf der letzten *for*-Schleife in Counting-Sort wird geändert zu

```
for j in range( 0, len(A) ):
```

Arbeitet das modifizierte Verfahren korrekt? Ist es stabil? Begründen Sie.

Aufgabe 3 (Binäre Tries) Punkte: 3

Geben Sie ein Verfahren an, das bei gegebener Menge $\mathcal{M}$ von Schlüsseln die exakte Höhe eines Binären Tries in $\mathcal{O}(n)$ ($n := |\mathcal{M}|$) berechnet, ohne den Baum aufzubauen. Gehen Sie von einer konstanten Anzahl Bits der Schlüssel aus.

Aufgabe 4 (Radix-Sort) Punkte: 5

Betrachten Sie MSD-Radix-Sort zur Basis 2. MSD (Most Significant Digit) bedeutet, daß zuerst nach dem höchstwertigen Bit sortiert wird (in der Vorlesung wurde zuerst nach dem niedrigsten Bit sortiert). Schreiben Sie ein Programm, welches zu einer gegebenen Menge $\mathcal{A}$ von N Zahlen die minimale Anzahl an Bits min_B berechnet, die zu deren Sortierung mit Radix-Sort betrachtet werden müssen, wenn

$$min_B := \sum_{0 \leq i < N} \sum_{0 \leq j < 32} bit(a_i, j) \ , \quad a_i \in \mathcal{A}$$

ist. Dabei gibt $bit(a_i, j)$ eine 1 zurück, wenn das j-te Bit der Zahl a_i wärend der Sortierung angeschaut (gelesen, verglichen,..) werden muss und eine 0 sonst. Gehen Sie dabei von 32-Bit-kodierten Integer-Zahlen aus. Verwenden Sie dabei zu keinem Zeitpunkt mehr als $\mathcal{O}(1)$ zusätzlichen Speicherplatz.
Tip: Entwerfen Sie einen rekursiven Algorithmus.