

Datenstrukturen und Algorithmen – Informatik I

6. Übung

Abgabe der Lösungen: 14. Juni 2004

Hinweis: Die Implementierungsaufgaben sind in Java oder C/C++ zu lösen. Der Quellcode sowie ein dokumentierter Beispiellauf sind beim Betreuer der Kleingruppe abzugeben.

Aufgabe 25: Programmieraufgabe: Quicksort

4+2+2 Punkte

- (a) Implementieren Sie den in der Vorlesung vorgestellten QuickSort Algorithmus. Dabei soll die Rekursion in

```
PROCEDURE QuickSort(l, r : CARDINAL) =  
  VAR k : CARDINAL;  
  BEGIN  
    IF l < r THEN  
      k := Partition(l, r);  
      QuickSort(l, k-1);  
      QuickSort(k+1, r);  
    END;  
  END QuickSort;
```

durch eine iterative Implementierung ersetzt werden. Geben Sie als Testlauf die Zwischenschritte und das Endergebnis beim Sortieren der Zahlenfolge aus Aufgabe 18, Übung 4 an.

- (b) Ändern Sie Ihre Implementierung so ab, dass das Pivot-Element in der Funktion `Partition` anders gewählt wird. Für jedes Teilfeld soll jeweils der Median¹ (das zweitkleinste Element) von drei Elementen (das erste, mittlere und letzte Element) als das Pivot-Element genommen werden.
- (c) Wieviele Vertauschungen bzw. Vergleiche benötigt QuickSort, um ein Feld mit n gleichen Elementen zu sortieren?

Aufgabe 26: Stabilität

3 Punkte

Untersuchen Sie Bubblesort, Shellsort und Quicksort auf Stabilität. Begründen Sie Ihre Antwort bzw. geben Sie ein Gegenbeispiel an.

Aufgabe 27: Programmieraufgabe: Suche nach dem Median

1+3 Punkte

Bei dem sogenannten „Auswahlproblem“ wird das Element mit dem k -kleinsten Schlüssel in einer n -elementigen Folge von Zahlen gesucht.

Formulieren Sie zwei Algorithmen, die das k -kleinste Element eines Arrays der Größe n finden: Der erste Algorithmus eignet sich für kleine k und benötigt etwa nk Schlüsselvergleiche; der zweite Algorithmus benötigt für sehr große n im Mittel $2n$ Vergleiche.

Begründen Sie argumentativ, dass die Algorithmen die angegebenen Komplexitätsanforderungen erfüllen.

Tip: Wandeln Sie zwei aus der Vorlesung bekannte Sortierverfahren geeignet ab.

Exkurs: Der Median einer n -elementigen Folge von Zahlen ist definiert als das $\lfloor \frac{n+1}{2} \rfloor$ -kleinste Element der Folge.

¹Die Definition des Medians ist in der Aufgabe 27 angegeben.

Aufgabe 28: HeapSort**2+1+1+1 Punkte**

- (a) Gegeben sei ein Feld mit dem Inhalt:

5	9	14	19	15	18	20	9	18	5	14
---	---	----	----	----	----	----	---	----	---	----

Überführen Sie dieses Feld manuell in einen binären Heap.

- (b) Statt mit binären Heaps kann man auch mit ternären Heaps arbeiten. In einem ternären Heap hat jedes Element des Feldes maximal drei Nachfolger. Geben Sie an, welche drei Feldelemente Nachfolger eines Elements mit Index j sind, und formulieren Sie die Heap-Eigenschaft für ternäre Heaps. Überführen Sie das Feld aus Aufgabenteil a) manuell in einen ternären Heap.
- (c) Erklären Sie kurz und prägnant, wie sich bei einem ternären Heap die Versicker-Operation ändert.
- (d) Wie wirkt sich der Übergang von binären zu ternären Heaps auf die Laufzeit von HeapSort aus? Geben Sie eine grobe Abschätzung.

Aufgabe 29: Sortieren in Blöcken**2+2 Punkte**

Betrachten Sie folgendes Sortierproblem: Gegeben seien ein Feld mit n Elementen und eine Blockgröße k . Wenn man sich das Feld in n/k Blöcke der Größe k aufgeteilt denkt (wir nehmen an, dass n/k eine natürliche Zahl ist), so soll für alle i, j mit $1 \leq i < j \leq n/k$ gelten, dass alle Elemente im i -ten Block kleiner sind als alle Elemente im j -ten Block. D.h., zum Sortieren muss nur noch innerhalb der Blöcke sortiert werden.

- (a) Zeigen Sie für dieses Sortierproblem eine obere Schranke von $O(n \log k)$ für die Anzahl benötigter Elementvergleiche, indem Sie die bekannte obere Schranke für das allgemeine Sortierproblem benutzen.
- (b) Zeigen Sie, dass die untere Schranke $\Omega(n \log k)$ ebenfalls gilt. Schätzen Sie dazu explizit die Anzahl der Permutationen ab, zwischen denen ein Algorithmus zur Lösung des obigen Sortierproblems unterscheiden muss. Tip: Verwenden Sie eine passende untere Schranke für die Fakultät (siehe Vorlesung).