Bergische Universität GH Wuppertal
Fachbereich Mathematik
Prof. Dr. A. Frommer, Holger Arndt

Informatik II: Algorithmen und Datenstrukturen

SS 2002

10. Übungsblatt

Aufgabe 29 (Indirektes Suchen und Sortieren, 6P)

a) Implementieren Sie eine Prozedur

```
void auswahlsort_indirekt (datensatz a[], int pi[], int n)

/* Bestimmt eine Permutation pi[0..n-1] der Zahlen
   0..n-1, so dass die Datensaetze nach aufsteigenden
   Schluesseln  sortiert sind:
   a[pi[0]].schluessel <= a[pi[1]].schluessel
                       <= ... <= a[pi[n-1]].schluessel */
```

die das Verfahren „Sortieren durch Auswahl“ indirekt durchführt.

b) Wandeln Sie die Funktion `binaere_suche` so in

```
int binaere_suche_indirekt(schluesseltyp s, datensatz a[],
                           int pi[], int anfang, int ende)

/* sucht in der Folge a[anfang..ende], die durch die Permutation
   pi[anfang..ende] indirekt geordnet ist, nach einem Datensatz
   Nr. i mit a[i].schluessel == s; gibt es einen solchen
   Datensatz, so wird sein Index i zurueckgeliefert, sonst
   anfang - 1                                                  */
```

ab, dass die Suche nach einem Element mit Schlüssel `s` in einem indirekt sortierten Feld durchgeführt wird.

c) Wenden Sie die zwei Routinen zum Test auf folgendes Beispiel an:

- Sortieren Sie die Datensätze mit den Schlüsseln 44, 55, 12, 42, 94, 18, 6, 67, 37, 13, 5, 18 indirekt und lassen Sie den Feldindex der Elemente mit den Schlüsseln 12, 5 und 43 ausgeben.

Aufgabe 30 (Modifiziertes Quicksort, 3T+3P)

a) Beim Aufruf eines Unterprogrammes bzw. einer Funktion wird Speicherplatz benötigt zum Ablegen lokaler Variablen und zum Abspeichern der Rücksprungadresse.
Zeigen Sie hiermit: Bei der in der Vorlesung vorgestellten Implementierung von Quicksort benötigt das Sortieren einer n-elementigen Folge im Worst Case $\mathcal{O}(n)$ zusätzlichen Speicherplatz.

b) Modifizieren Sie die Prozedur, so dass nur noch $\mathcal{O}(\log n)$ zusätzlicher Speicherplatz im Worst Case benötigt wird (mit Begründung). Testen Sie Ihre Implementierung geeignet.

Hinweis: Die Prozedur enthält dann nur noch einen rekursiven Aufruf, dafür aber eine zusätzliche Schleife.

Aufgabe 31 (Optimiertes Bubblesort, 4T+2P)

Beim Bubblesort-Verfahren, wie es in der Vorlesung vorgestellt wurde, lässt sich folgendes beobachten. Wird bei einem Durchlauf der i-Schleife der letzte Tausch mit den Datensätzen a_l und a_{l+1} vorgenommen, so sind alle rechts von a_l stehenden Datensätze bereits fertig sortiert.

a) Formulieren Sie einen Algorithmus *Optimiertes Bubblesort*, der diese Beobachtung ausnutzt.

b) Wie ändert sich der Aufwand im Best Case und im Worst Case verglichen mit dem ursprünglichen Algorithmus?

c) Implementieren Sie den Algorithmus aus Teil a).

Abgabe: Mi., 26.06.2002, 14 Uhr