

Datenstrukturen und Algorithmen – Informatik I

5. Übung

Abgabe der Lösungen: 7. 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 20: Laufzeitverhalten von Sortierverfahren

3+2+3 Punkte

- (a) Bestimmen Sie mit den Verfahren aus der Vorlesung (Kapitel 1.2) das Laufzeitverhalten der Sortierverfahren *SelectionSort*, *InsertionSort* und *BubbleSort*. Tragen Sie die Ergebnisse in eine Tabelle ein:

	best		average		worst	
	moves	compares	moves	compares	moves	compares
<i>SelectionSort</i>						
<i>InsertionSort</i>						
<i>BubbleSort</i>						

- (b) Welchen dieser Sortieralgorithmen würden Sie bei folgenden Problemen bevorzugen? Begründen Sie Ihre Wahl.

- (a) Sortieren eines Feldes mit Elementen großer Record-Länge.
- (b) Sortieren eines Feldes, das aus einem großen sortierten und einem kleinen unsortierten Bereich am Ende des Feldes besteht.

- (c) Finden und implementieren Sie eine Variante von *BubbleSort*, die für fast sortierte Felder eine Laufzeitkomplexität von $O(n)$ hat. Verwenden Sie folgende Testdaten:

```

1 2 3 4 5 6 7 8 9 10 11 12 13
2 4 6 8 10 12 1 3 5 7 9 11 13
13 12 11 10 9 8 7 6 5 4 3 2 1

```

Aufgabe 21: Implementierung: Sortieren

2+2+2+2 Punkte

Implementieren Sie

- (a) *SelectionSort*
- (b) *InsertionSort*
- (c) *BubbleSort*
- (d) *ShellSort* (siehe Aufgabe 22)

und testen diese an den Daten aus Aufgabe 18 (b).

Aufgabe 22: Shellsort**3 Punkte**

Ein von *BubbleSort* abgeleitetes Sortierverfahren ist *ShellSort*. Zur Verbesserung des Laufzeitverhaltens werden Elemente auch über größere Entfernungen vertauscht. Eine entsprechende Prozedur ist:

```
PROCEDURE ShellSort(a : ARRAY[1..n] of ItemType)
BEGIN
  schrittweite=n DIV 2 # initiale Schrittweite
  WHILE schrittweite > 0 DO
    FOR i= schrittweite+1 TO n DO
      j:=i - schrittweite
      WHILE j>0 DO
        IF a[j] > a[j+schrittweite] THEN
          tausche(a[j],a[j+schrittweite]);
          j:=j-schrittweite;
        ELSE
          j:=0;
        ENDIF
      ENDWHILE
    ENDFOR
    schrittweite=schrittweite DIV 2;
  ENDWHILE
END
```

Untersuchen Sie manuell den Lauf von *ShellSort* auf der Eingabe

1 7 3 2 0 5 0 8.

Geben Sie die Zwischenergebnisse nach allen Vertauschungen an.

Aufgabe 23: Mindestkomplexität**3 Punkte**

Zeigen Sie, dass jeder Sortieralgorithmus, der bei gegebenem Feld der Größe n nur dadurch Feldelemente bewegt, dass er benachbarte Elemente vertauscht, zumindest einen (worst-case) Zeitaufwand von $\Omega(n^2)$ hat.

Aufgabe 24: 3-Wert-Sortieren**4 Punkte**

In einem Array seien Records mit den Schlüsselwerten rot, grün, blau enthalten. Es soll eine Folge dieser Records hergestellt werden, in der am Anfang alle roten, dann all grünen und zuletzt alle blauen Schlüssel vorkommen. Als Operationen seien nur das Feststellen eines Schlüsselwertes und das Vertauschen zweier Records erlaubt. Geben Sie einen Algorithmus an, der das Array in situ mit Laufzeitkomplexität $O(n)$ sortiert.

Zeigen Sie, dass Ihr Algorithmus mit $O(n)$ Vertauschungen das Array sortiert.