

Prof. Dr. Frank Noé  
Dr. Christoph Wehmeyer  
Tutoren:  
Katharina Colditz; Anna Dittus;  
Felix Mann; Christopher Pütz

## 7. Übung zur Vorlesung Computerorientierte Mathematik I

Lösung

### Aufgabe 1 (*Komplexität, 2T*):

Sei  $\mathbf{A} \in \mathbb{R}^{n \times n}$  eine quadratische Matrix und  $\mathbf{x} \in \mathbb{R}^n$  ein Vektor. Zeigen Sie, dass die Komplexität der Matrix-Vektor-Multiplikation  $\mathbf{y} = \mathbf{Ax}$  höchstens quadratisch in der Dimension  $n$  ist. Dabei sei das Aufwandsmaß die Anzahl der nötigen Multiplikationen und Additionen.

#### Lösung

Wir zeigen, dass der Standardalgorithmus zur Matrix-Vektor-Multiplikation quadratischen Aufwand besitzt. Damit kann die Komplexität des Problems nur noch besser sein. Der normale Algorithmus berechnet  $n$  Skalarprodukte zwischen einer Zeile der Matrix  $\mathbf{A}$  und  $\mathbf{x}$ . Für ein Skalarprodukt benötigt man  $n$  Multiplikationen und  $n - 1$  Additionen. Damit erhalten wir also  $n(2n - 1) = \mathcal{O}(n^2)$  Operationen.

### Aufgabe 2 (*Sortieren, 8T*):

Ein weiterer Sortieralgorithmus ist **Insertion Sort**. Dieses Verfahren lässt sich folgendermaßen beschreiben (Eingabe sei eine Liste  $\mathbf{x}$  der Länge  $n \geq 2$ , wie gewohnt bezeichnen wir mit  $\mathbf{x}(i)$  das  $i$ -te Element in der Liste):

1. Setze  $i = 2$ .
  2. Setze  $j = i$ . Solange  $j > 1$  und  $\mathbf{x}(j - 1) > \mathbf{x}(j)$  gelten, wiederhole:
    - (a) Vertausche  $\mathbf{x}(j - 1)$  und  $\mathbf{x}(j)$ .
    - (b) Setze  $j = j - 1$ .
  3. Falls  $i = n$  ist, breche ab. Sonst setze  $i = i + 1$  und wiederhole Schritt 2.
- a) (*3T*) Begründen Sie, dass Insertion Sort korrekt ist, also dass am Ende die Liste  $\mathbf{x}$  aufsteigend sortiert ist.
- b) (*3T*) Zeigen Sie, dass Insertion Sort quadratische Laufzeit besitzt, also dass

$$T_A(n) = \mathcal{O}(n^2).$$

Dabei misst die Eingabegröße  $n$  die Länge der Liste  $\mathbf{x}$  und das Aufwandsmaß ist die Anzahl der benötigten Vergleiche.

c) (2T) Sei  $\mathbf{x}$  eine Liste der Länge  $n$ , die bereits aufsteigend sortiert ist. Wie viele Vergleiche benötigt Insertion Sort, und wie viele benötigt Bubble Sort?

### Lösung

a) Insertion Sort ist korrekt: Man zeigt für alle  $i \geq 2$ , dass nach der Ausführung von Schritt 2 die ersten  $i$  Elemente der Liste aufsteigend sortiert sind. Am besten macht man das per Induktion: Für den Induktionsanfang  $i = 2$  beobachten wir, dass durch Schritt 2 die ersten beiden Elemente in die richtige Reihenfolge gebracht werden. Sei nun die Liste bis zum  $i$ -ten Element sortiert. Dann bewirkt Schritt 2, dass das Element  $\mathbf{x}(i+1)$  solange nach links verschoben wird, bis ein kleineres (oder gleiches) Element gefunden ist. Da die ersten  $i$  Einträge sortiert waren, sind nun die ersten  $i+1$  Einträge sortiert. Damit sind am Ende alle  $n$  Einträge sortiert. (3 Punkte).

b) Zuerst beobachten wir, dass Schritt 2  $n-1$ -mal wiederholt wird. Im schlimmsten Fall ist das Element  $\mathbf{x}(i)$  immer kleiner als alle vorigen Elemente, und in Schritt 2 müssen  $i-1$  Vergleiche durchgeführt werden. Damit erhalten wir

$$\begin{aligned} T_A(n) &= 1 + 2 + \dots + n - 1 \\ &= \frac{1}{2}n(n-1) \\ &= \mathcal{O}(n^2). \end{aligned}$$

(3 Punkte)

c) Das ist der optimale Fall für Insertion Sort. In diesem Fall führt Schritt 2 immer nur einen Vergleich aus und es werden insgesamt nur  $n-1 = \mathcal{O}(n)$  Vergleiche durchgeführt. Bubble Sort hingegen führt immer alle  $i-1$  Vergleiche aus und benötigt daher  $\frac{1}{2}n(n-1) = \mathcal{O}(n^2)$  Vergleiche. (2 Punkte)