Algorithmen und Datenstrukturen, Übung 1

Marouane Soussi , Lars Happel

April 2022

Aufgabe 1

siehe .java Datei

Aufgabe 2 - Insertionsort

a)

Für $n = 4$ sind es 4 Vergleiche. Für größere n: $2n - 2$ Vergleiche, da zuerst die 2 am Ende des Array nach vorn geschoben wird, allerdings das Array vorn bei 2 anfängt und somit nicht ganz n Vergleiche benötigt werden. Anschließend wird noch die 1 nach vorn geschoben, vor die beiden Zweien.

b)

$\frac{n}{2}$ Vergleiche. Das Array besteht aus Zahlenpaaren die an sich in verkehrter Reihenfolge sind. Allerdings sind beide Zahlen von Paar 1 kleiner als beide aus Paar usw. Es müssen daher in jedem Zahlenpaar einmal die beiden Zahlen vertauscht werden.

c)

$\sum_{i=1}^{\frac{n}{2}-1} i$ Vergleiche. Begründung:
Für $n = 4$ wäre das Array: [1, 3, 1, 4], d.h. 2 Vergleiche.
Für $n = 8$ wäre das Array: [1, 5, 1, 6, 1, 7, 1, 8]
Dies sind $1 + 2 + 3$ Vergleiche, da die 1 immer nach links verschoben werden muss. Für weitere n bildet sich das Muster $1 + 2 + 3 + \ldots \frac{n}{2} - 2$ Vergleiche ($\sum_{i=1}^{\frac{n}{2}-1} i$)

Aufgabe 3 - Senken zählen

a) Pseudocode

Input: a[1, ..., n] mit n ¿ 3 Result: Anzahl von Senken

```
c := 0
for i=2 to n-1 do:
  if (a[i] < a[i-1] and a[i] < a[i+1] do:
    c := c+1
return c
```

b) Schleifeninvariante

Zu Beginn jeder Iteration der Schleife ist in der Variable c die bisher gefundene Anzahl an Senken gespeichert

Initialization:

Für $n = 2$ wird zwischen $a[1] > a[2] < a[3]$. Falls die Voraussetzung erfüllt ist, wird c inkrementiert.

Maintenance:

In jeder Iteration gilt die Schleifeninvariante. Jedes Mal wird c inkrementiert, falls $a[i-1] > a[i]$ und $a[i+1] > a[i]$.

Termination:

Die Schleife terminiert immer mit $i = n$. An dieser Stelle ist c die Anzahl der Senken im Array.

c) Weniger als n-2 Schritte

Da keine 2 nebeinanderliegenden Positionen beide Senken sein können, lässt sich sobald man eine Senken gefunden hat ausschließen, dass das Feld rechts davon auch eine Senke ist. Man darf also jedes Mal wenn eine Senke gefunden wurde eine Schleifeniteration überspringen:

```
c := 0
for i=2 to n-1 do:
  if (a[i] < a[i-1] and a[i] < a[i+1] do:
    c := c+1
    i := i+1 // Überspringen einer Iteration
return c
```