Sortierverfahren Lösungen+ noch mehr Prüfungsvorbereitung

Aufgabe 1

Selectionsort					Vergleiche	Swaps
7	9	1	3	5	4	1
1	9	7	3	5	3	1
1	3	7	9	5	2	1
1	3	5	9	7	1	1
1	3	5	7	9		

Aufgabe 2

Insertionsort					Vergleiche	Shifts
8	7	9	3	4	1	1
7	8	9	3	4	1	0
7	8	9	3	4	3	3
3	7	8	9	4	4	3
3	4	7	8	9		

Aufgabe 3

Bubblesort					Vergleiche	Swaps
6	4	5	2	3	1	1
4	6	5	2	3	1	1
4	5	6	2	3	1	1
4	5	2	6	3	1	1
4	5	2	3	6	1	0
4	5	2	3	6	1	1
4	2	5	3	6	1	1
4	2	3	5	6	1	1
2	4	3	5	6	1	1
2	3	4	5	6	1	0
2	3	4	5	6		

Aufgabe 4

(a)

QS-Partition					Vergleiche	Swaps
2	7	8	6	**5**		
2	7	8	6	**5**		
2	7	8	6	**5**		
2	7	8	6	**5**		
2	7	8	6	**5**		
2	**5**	8	6	7		

(b) Die Partitionierungsfunktion gibt den Index $40 + 1 = 41$ zurück.

Aufgabe 5

Selectionsort mit einer absteigend sortierten Liste mit 8 Elementen.

Anzahl Vergleiche: $7 + 6 + \cdots + 2 + 1 = \frac{7}{2} \cdot (1 + 7) = 28$

Anzahl Vertauschungen: $1 + 1 + \cdots + 1 + 1 = 8 - 1 = 7$

Zur Illustration ein Beispiel mit nur $n = 4$ Elementen:

Selectionsort				Vergleiche	Swaps
4	3	2	1	3	1
1	3	2	4	2	1
1	2	3	4	1	1
1	2	3	4		

Aufgabe 6

Insertionsort mit einer aufsteigend sortierten Liste mit 8 Elementen.

Anzahl Vergleiche: $1 + 1 + \cdots + 1 + 1 = 8 - 1 = 7$

Anzahl Verschiebungen: $0 + 0 + \cdots + 0 + 0 = 0$

Zur Illustration ein Beispiel mit nur $n = 4$ Elementen:

Insertionsort				Vergleiche	Verschiebungen
1	2	3	4	1	0
1	2	3	4	1	0
1	2	3	4	1	0
1	2	3	4		

Aufgabe 7

Bubblesort mit einer absteigend sortierten Liste mit 8 Elementen.

Anzahl Vergleiche: $7 + 6 + \cdots + 2 + 1 = \frac{7}{2} \cdot (1 + 7) = 28$

Anzahl Vertauschungen: $7 + 6 + \cdots + 2 + 1 = \frac{7}{2} \cdot (1 + 7) = 28$

Zur Illustration ein Beispiel mit nur $n = 4$ Elementen:

Bubblesort				Vergleiche	Vertauschungen
4	3	2	1	1	1
3	4	2	1	1	1
3	2	4	1	1	1
2	3	1	4	1	1
2	3	1	4	1	1
2	1	3	4	1	1
1	2	3	4		

Aufgabe 8

Da jedes Element vor dem Pivotelement mit dem Pivotelement verglichen wird, gibt es bei 29 Elementen 28 Vergleiche.

Aufgabe 9

(a) Selectionsort

```
def sort(A):
    n = len(A)
    for i in range(0, n-1):
        k = i
        for j in range(i+1, n):
            if A[j] < A[k]:
                k = j
        A[k], A[i] = A[i], A[k]
```

(b) Bubblesort

```
def sort(A):
    n = len(A)
    for i in range(0, n-1):
        for j in range(0, n-i-1):
            if A[j] > A[j+1]:
                A[j], A[j+1] = A[j+1], A[j]
```

Aufgabe 10

(a) Bubblesort

3	4	2	1	5
3	2	4	1	5
3	2	1	4	5
2	3	1	4	5
2	1	3	4	5

(b) Insertionsort

3	4	2	1	5
1	4	2	3	5
1	2	4	3	5
1	2	3	4	5
1	2	3	4	5

(c) Selectionsort

3	4	2	1	5
1	4	2	3	5
1	2	4	3	5
1	2	3	4	5
1	2	3	4	5

Aufgabe 11

	Worst Case	Best Case
Quicksort	$C \cdot n^2$	$C \cdot n \cdot \log_2(n)$
Bubblesort	$C \cdot n^2$	$C \cdot n^2$
Insertionsort	$C \cdot n^2$	$C \cdot n$

Aufgabe 12

Gegeben: $C \cdot 100^2 = 0.2$

Gesucht: $C \cdot 200^2 = ?$

$t = C \cdot 200^2 = C \cdot (2 \cdot 100)^2 = C \cdot 2^2 \cdot 100^2 = 4 \cdot C \cdot 100^2 = 4 \cdot 0.2\,\mathrm{s} = 0.8\,\mathrm{s}$

Aufgabe 13

Der Quicksort-Algorithmus hat die Worst Case-Laufzeit, wenn die zu sortierende Liste bereits auf- oder absteigend sortiert ist.

Das folgende Beispiel zeigt, dass Quicksort auf einer aufsteigend sortierten Liste jewils nur ein Element an die richtige Position setzt – nämlich das letzte. Die Situation bei einer absteigend sortierten Liste ist etwas komplizierter, verhält sich aber ähnlich. Der hervorgehobene Ausdruck bei den Vertauschungen zeigt an, dass das Pivotelement an seine richtige Position getauscht wird (hier natürlich am gleichen Platz).

Quicksort				Vergleiche	Swaps
1	2	3	4	1	1
1	2	3	4	1	1
1	2	3	4	1	1
1	2	3	4	0	**1**
1	2	3		1	1
1	2	3		1	1
1	2	3		0	**1**
1	2			1	1
1	2			0	**1**
1					

Aufgabe 14

- *Randomized Quicksort:* Wähle aus den Elementen vor dem Pivotelement eines zufällig aus und vertausche es mit dem Element an der Pivotposition.
- *Median of three:* Bestimme den Median des ersten, mittleren und letzten Elements und vertausche ihn mit dem Element an der Pivotposition.

Aufgabe 15

`[24, 27, 13, 23, 22, 21, 1]`

`[1, 13, 21, 22, 23, 24, 27]` (sortiert)

Median: 22 (Synonym: *Zentralwert*)

Aufgabe 16

`[20, 16, 27, 11, 8, 24, 26, 3]`

`[3, 8, 11, 16, 20, 24, 26, 27]` (sortiert)

Median: 18 (Synonym: *Zentralwert*)

Aufgabe 17

```
def is_sorted(A):
    for i in range(0, len(A)-1):
        if A[i] > A[i+1]:
            return False
    return True
```