Aufgabe 1

Blau: Element an der korrekten Position

Rot: das minimale Element der unsortierten Teilliste

9	33	6	8	7	Vergleiche	Vertauschungen
3	9	66	8	7	4	1
3	**6**	9	8	77	3	1
3	**6**	**7**	88	9	2	1
3	**6**	**7**	**8**	9	1	1
					10	**4**

- Auch wenn ein Element (wie die 8) bereits an der richtigen Position steht, führt der Algorithmus eine Vertauschung (mit sich selbst) durch. Um das zu verhindern, müsste man vor bei jeder Vertauschung testen, ob eine Vertauschung nötig ist, was am Ende sogar mehr Zeit kosten würde.
- Das letzte Element muss nicht mehr einsortiert werden, da es das grösste Element sein musss. Daher genügt es, die äussere for-Schleife nur bis zum zweitletzten Element laufen zu lassen.

Aufgabe 2

Insertionsort

Blau: relativ sortierte Teilliste

Rot: das nächste einzusortierende Element

9	**3**	6	8	7	Vergleiche	Verschiebungen
3	**9**	**6**	8	7	1	1
3	**6**	**9**	**8**	7	2	1
3	**6**	**8**	**9**	**7**	2	1
3	**6**	**7**	**8**	**9**	3	2
					8	**13**

Aufgabe 3

Bubblesort

Blau: Element an der korrekten Position

Rot: Vertauschungskandidaten

					Vergleiche	Vertauschungen
9	**3**	6	8	7		
3	**9**	**6**	8	7	1	1
3	6	**9**	**8**	7	1	1
3	6	8	**9**	**7**	1	1
3	**6**	8	7	**9**	1	1
3	**6**	**8**	7	**9**	1	0
3	6	**8**	**7**	**9**	1	0
3	**6**	7	**8**	**9**	1	1
3	**6**	**7**	**8**	**9**	1	0
3	**6**	**7**	**8**	**9**	1	0
3	**6**	**7**	**8**	**9**	1	0
					10	**5**

Aufgabe 4

(a)

...	2	8	3	9	4	**6**	...
...	**2**	8	3	9	4	**6**	...
...	**2**	**8**	3	9	4	**6**	...
...	**2**	**3**	**8**	9	4	**6**	...
...	**2**	**3**	**8**	**9**	4	**6**	...
...	**2**	**3**	**4**	**9**	**8**	**6**	...
...	**2**	**3**	**4**	**6**	**8**	**9**	...

(b)

99	100	101	102	103	104	105	106
...	...	...	...	...	...	...	...
...	**2**	**3**	**4**	**6**	**8**	**9**	...

Der Partitionierungsschritt gibt den Index 103 zurück.

Damit wird den folgenden rekursiven Aufrufen mitgeteilt, das noch die Teillisten mit den Indizes 100–102 und 104–105 sortiert werden müssen.

Aufgabe 5

8	4	7	1	3	9	**5**
4	8	7	1	3	9	**5**
4	1	7	8	3	9	**5**
4	1	3	8	7	9	**5**
4	1	3	**5**	7	9	8
4	1	**3**				
1	4	**3**				
1	**3**	4				
1						
		4				
				7	9	**8**
				7	9	**8**
				7	**8**	9
				7		
						9

Aufgabe 6

Selectionsort:

(a) die Elemente sind aufsteigend sortiert:

- $6 + 5 + 4 + 3 + 2 + 1 = 21$ Vergleiche
- $1 + 1 + 1 + 1 + 1 + 1 = 6$ Vertauschungen

(b) die Elemente sind absteigend sortiert:

- $6 + 5 + 4 + 3 + 2 + 1 = 21$ Vergleiche
- $1 + 1 + 1 + 1 + 1 + 1 = 6$ Vertauschungen

Aufgabe 7

Insertionsort

(a) die Elemente sind aufsteigend sortiert:

- $1 + 1 + 1 + 1 + 1 + 1 = 6$ Vergleiche
- $0 + 0 + 0 + 0 + 0 + 0 = 0$ Verschiebungen

(b) die Elemente sind absteigend sortiert:

- $1 + 2 + 3 + 4 + 5 + 6 = 21$ Vergleiche
- $1 + 2 + 3 + 4 + 5 + 6 = 21$ Verschiebungen

Aufgabe 8

Selectionsort:

(a) die Elemente sind aufsteigend sortiert:

- $6 + 5 + 4 + 3 + 2 + 1 = 21$ Vergleiche
- $0 + 0 + 0 + 0 + 0 + 0 = 0$ Vertauschungen

(b) die Elemente sind absteigend sortiert:

- $6 + 5 + 4 + 3 + 2 + 1 = 21$ Vergleiche
- $6 + 5 + 4 + 3 + 2 + 1 = 21$ Vertauschungen

Aufgabe 9

19 Vergleiche, da die ersten 19 Elemente jeweils mit dem Pivotelement verglichen werden müssen.

Aufgabe 10

```python
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]
```

Es handelt sich um Bubblesort.

Aufgabe 11

```python
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]
```

Es handelt sich um Selectionsort.

Aufgabe 12

(a) Insertionsort

(b) Quicksort

(c) Selectionsort

Aufgabe 13

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

Aufgabe 14

$C \cdot 1000^2 = 0.1\,\mathrm{s}$

$C \cdot 7000^2 = C \cdot (7 \cdot 1000)^2 = 49 \cdot C \cdot 1000^2 = 49 \cdot 0.1\,\mathrm{s} = 4.9\,\mathrm{s}$

Aufgabe 15

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

Aufgabe 16

- *Randomized Quicksort:* Wähle ein zufälliges Element.
- *Median of three:* Wähle den Median des ersten, mittleren und letzten Elements.

Aufgabe 17

(a) `[8, 12, 27, 3, 16, 29, 17]`

sortieren: `[3, 8, 12, 16, 17, 27, 29]`

Median: 16

(b) `[23, 14, 7, 9, 55, 6, 2, 12]`

sortieren: `[2, 6, 7, 9, 12, 14, 23, 55]`

Median: 10.5

Aufgabe 18

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

Aufgabe 19

```python
def findMinPos(A):
    minPos = 0
    minElement = A[0]
    for i in range(1, len(A)):
        if A[i] < minElement:
            minElement = A[i]
            minPos = i
    return minPos
```