Aufgabe 1

Gnomesort

3	5	1	6	
3	5	1	6	
3	5	1	6	
3	1	5	6	
1	3	5	6	
1	3	5	6	
1	3	5	6	
1	3	5	6	
1	3	5	6	

Aufgabe 2

```
def gnomesort(L):
    i = 0
    while i<len(L):
        if i==0:
            i=1
        elif A[i-1]>A[i]:
            A[i-1],A[i]=A[i],A[i-1]
            i=i-1
        else:
            i=i+1
```

Aufgabe 3

Komplexität von Gnomesort:

(a) Best Case: aufsteigend sortiertes Array $\Rightarrow O(n)$

(b) Worst Case: absteigend sortiertes Array $\Rightarrow O(n^2)$

Aufgabe 4

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 5

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

Aufgabe 6

Komplexität von Selectionsort

Selectionsort hat immer $O(n^2)$

Aufgabe 7

Insertionsort

					Vergleiche	Verschiebungen
7	9	1	3	5	1	0
7	9	1	3	5	2	3
1	7	9	3	5	3	3
1	3	7	9	5	3	3
1	3	5	7	9		

Aufgabe 8

```
def insertionsort(L):
    n = len(A):
    for i in range(0,n):
        x = A[i]
        j = i - 1
        while(j>=0 and A[j]>x):
            A[j+1] = A[j]
            j = j - 1
        A[j+1] = x
```

Aufgabe 9

Komplexitäv von Insertionsort:

(a) Best Case: aufsteigend sortiertes Array $\Rightarrow O(n)$

(b) Worst Case: absteigend sortiertes Array $\Rightarrow O(n^2)$

Aufgabe 10

Quicksort

unterstrichen: Pivotelement; rot: Element an richtiger Position

9	5	1	8	7	4	6
5	9	1	8	7	4	6
5	1	9	8	7	4	6
5	1	4	8	7	9	6
5	1	4	6	7	9	8
1	5	4		7	9	8
1	4	5		7	9	8
1		5		7	9	8
		5		7	9	8
				7	8	9
				7		9
						9

Aufgabe 11

Komplexität von Quicksort:

(a) Im Best Case teilt das Pivotelement nach dem Partionierungsschritt den Rest der Liste jeweils in etwa zwei gleich grosse Hälften. $\Rightarrow O(n \log n)$

(b) Im Worst Case ist das Pivotelement immer das grösste oder kleinste Element der aktuellen Teilliste, der Länge k, so dass nach jedem Partitionierungsschritt immer noch $(k-1)$ Elemente sortiert werden müsssen. $\Rightarrow O(n^2)$

Aufgabe 12

Quicksort tritt zum Beispiel dann auf wenn die vor dem Sortieren bereits auf- oder absteigend sortiert ist. Gegenmassnahmen:

- *Zufällige Pivot-Wahl:* Vertausche ein zufällig gewähltes Element aus der zu sortierenden Teilliste mit dem Element an der Pivotposition (am Ende der Teilliste). Dies reduziert die Gefahr, dass das Pivot-Element immer das kleinste oder grösste Element ist.
- *Median-of-Three-Strategie:* Wähle das Pivot-Element als Median aus einer Gruppe von drei Elementen (z. B. dem ersten, dem letzten und dem mittleren Element). Dies hilft, ein besseres Pivot-Element zu finden und die Teilung der Daten zu optimieren.

Aufgabe 13

```
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 14

```
def sort(L):
    n = len(L)
    i = 0
    while i < n:
        if i == 0:
            i = i + 1
        elif L[i-1] > L[i]:
            L[i], L[i-1] = L[i-1], L[i]
            i = i - 1
        else:
            i = i + 1
```

Es handelt sich um Gnomesort.

Aufgabe 15

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

Es handelt sich um Insertionsort.

Aufgabe 16

```
def sort(A):
    sort_helper(A, 0, len(A))

def helper(A, a, b):
    if a < b:
        m = function(A, a, b)
        helper(A, a, m)
        helper(A, m+1, b)
```

```
def function(A, a, b):
    p = A[b-1]
    i = a-1
    for j in range(a, b-1):
        if A[j] <= p
            i += 1
            A[i], A[j] = A[j], A[i]
    A[i+1], A[b-1] = A[b-1], A[i+1]
    return i+1
```

Es handelt sich um Quicksort. (OK, das war einfach!)

Aufgabe 17

7	2	4	3	8	5
2	7	4	3	8	5
2	3	4	7	8	5
2	3	4	5	8	7
2	3	4	5	7	8

Selectionsort

7	2	4	3	8	5
2	7	4	3	8	5
2	4	7	3	8	5
2	4	3	7	8	5
2	4	3	5	8	7

Quicksort

7	2	4	3	8	5
2	7	4	3	8	5
2	4	7	3	8	5
2	4	3	7	8	5
2	3	4	7	8	5

Gnomesort

Aufgabe 18

4	3	8	1	9	5
3	4	8	1	9	5
3	4	1	8	9	5
3	1	4	8	9	5
1	3	4	8	9	5

Gnomesort

4	3	8	1	9	5
1	3	8	4	9	5
1	3	4	8	9	5
1	3	4	5	9	8
1	3	4	5	8	9

Selectionsort

4	3	8	1	9	5
3	4	8	1	9	5
1	3	4	8	9	5
1	3	4	5	8	9

Insertionsort

Aufgabe 19

$T(n) \in O(n^2) \quad \Rightarrow \quad T(n) = C \cdot n^2$

$T(10^4) = C \cdot (10^4)^2 = 6\,\mathrm{s}$

$T(10^5) = C \cdot (10^5)^2 = 10^2 \cdot C \cdot (10^4)^2 = 100 \cdot 6\,\mathrm{s} = 600\,\mathrm{s} = 10\,\mathrm{min}$

Aufgabe 20

$T(n) \in O\big(n \log(n)\big) \quad \Rightarrow \quad T(n) = C \cdot n \log(n)$

$T(10^4) = C \cdot 10^4 \cdot \log(10^4) \overset{*}{=} 6\,\mathrm{s}$

$$T(10^5) = C \cdot 10^5 \cdot \log(10^5) = C \cdot 10 \cdot 10^4 \cdot \tfrac{5}{4} \cdot \log(10^4)$$
$$= \tfrac{50}{4} \cdot \underbrace{C \cdot 10^4 \cdot \log(10^4)}_{6\,\mathrm{s}} \overset{*}{=} \tfrac{50}{4} \cdot 6\,\mathrm{s} = 75\,\mathrm{s}$$

Beachte: $\log(x^r) = r \cdot \log(x) \Rightarrow \log(10^5) = \log\left(10^4\right)^{\frac{5}{4}} = \frac{5}{4} \cdot \log(10^4)$

oder mit der passenden Basis $\log(x) = \log_{10}(x)$:

$T(10^4) = C \cdot 10^4 \cdot \log_{10}(10^4) = C \cdot 10^4 \cdot 4 = 6\,\mathrm{s}$

$T(10^5) = C \cdot 10^5 \cdot \log_{10}(10^5) = C \cdot 10^5 \cdot 5 = 10 \cdot \frac{5}{4} \cdot \underbrace{C \cdot 10^4 \cdot 4}_{6\,\mathrm{s}} = \frac{50}{4} \cdot 6\,\mathrm{s} = 75\,\mathrm{s}$