Aufgabe 1

Zeige schrittweise, wie Selectionsort die Liste `L = [7, 9, 1, 3, 5]` aufsteigend sortiert. Notiere für jeden Schritt die Anzahl der Vergleiche und der Vertauschungen.

Aufgabe 2

Zeige schrittweise, wie Insertionsort die Liste `L = [8, 7, 9, 3, 4]` aufsteigend sortiert. Notiere für jeden Schritt die Anzahl der Vergleiche und der Verschiebungen.

Aufgabe 3

Zeige schrittweise, wie Bubblesort die Liste `L = [6, 4, 5, 2, 3]` aufsteigend sortiert. Notiere für jeden Schritt die Anzahl der Vergleiche und der Vertauschungen.

Aufgabe 4

(a) Zeige schrittweise, wie die Partitionierungfunktion von Quicksort den folgenden Listenausschnitt verarbeitet.

`..., 2, 7, 8, 6, 5, ...`

Protokolliere dein Vorgehen, indem du jeweils eine neue Zeile beginnst, wenn ein Element (evtl. auch an Ort und Stelle) getauscht wird. Es kann aber auch jeder Einzelschritt angegeben werden.

(b) Welchen Wert gibt der Partitionierungsschritt zurück, wenn das erste Element (ganz links) der obigen Liste den Index $i = 40$ hat?

Aufgabe 5

Wie viele Vergleiche und wie viele Vertauschungen benötigt Selectionsort, um eine absteigend sortierte Liste mit 8 Elementen aufsteigend zu sortieren?

Aufgabe 6

Wie viele Vergleiche und wie viele Verschiebungen benötigt Insertionsort um eine bereits aufsteigend sortierte Liste mit 8 Elementen aufsteigend zu sortieren?

Aufgabe 7

Wie viele Vergleiche und wie viele Vertauschungen benötigt Bubblesort um eine absteigend sortierte Liste mit 8 Elementen aufsteigend zu sortieren?

Aufgabe 8

Wie viele Vergleiche benötigt die Partitionierungsfunktion von Quicksort für eine Teilliste mit 29 Elementen?

Aufgabe 9

Welcher Sortieralgorithmus wird mit den folgenden Python-Programmen implementiert?

(a)
```
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)
```
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

Bestimme anhand der ersten 4 Verarbeitungsschritte, um welchen Sortieralgorithmus es sich handelt und gib eine kurze Begründung. Beachte:

- Nicht alle Algorithmen sortieren die Liste in 4 Schritten bis zum Ende.
- Es werden nur Schritte dargestellt, die *potenziell* Elemente vertauschen oder verschieben. Es ist aber möglich, dass ein Element mit sich selbst vertauscht oder um null Positionen verschoben wird.
- Ein Algorithmus wird höchstens einmal verwendet.

(a)

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)

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

(c)

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

Gib die ungefähre Sortierdauer der von uns behandelten Sortieralgorithmen für den besten und den schlechtesten Fall in Abhängigkeit der Arraylänge n und mit einer Proportionalitätskonstanten C an.

	Worst Case	Best Case
Quicksort		
Bubblesort		
Insertionsort		

Aufgabe 12

Ein Programm, das Arrays mit der Laufzeit $C \cdot n^2$ sortiert, benötigt auf einem Computer ungefähr 0.2 Sekunden, um eine Liste mit 100 Zufallszahlen zu sortieren. Wie lange wird es ungefähr dauern, wenn dasselbe Programm auf dem gleichen Computer eine Liste mit 200 Zufallszahlen sortiert?

Aufgabe 13

Beschreibe, für welche Art von Listen der Quicksort-Algorithmus die Worst Case-Laufzeit aufweist?

Aufgabe 14

Beschreibe zwei Pivotstragegien für das Quicksort-Verfahren, um die Worst Case-Laufzeit zu verhindern.

Aufgabe 15

Berechne den Median der Listenwerte `[24, 27, 13, 23, 22, 21, 1]`

Aufgabe 16

Berechne den Median der Listenwerte `[20, 16, 27, 11, 8, 24, 26, 3]`

Aufgabe 17

Schreibe eine Python-Funktion `is_sorted(A)`, mit einer Liste `A` als Argument, die `True` zurückgibt, wenn die Liste bereits sortiert ist und `False` sonst.