

Aufgabe 1

Selectionsort:

8	3	7	6	2	4	Vergleiche	Vertauschungen
2	3	7	6	8	4	5	1
2	3	7	6	8	4	4	1
2	3	4	6	8	7	3	1
2	3	4	6	8	7	2	1
2	3	4	6	7	8	1	1
						15	5

Aufgabe 2

Selectionsort:

9	3	5	2	1	4	8	Vergleiche	Vertauschungen
1	3	5	2	9	4	8	6	1
1	2	5	3	9	4	8	5	1
1	2	3	5	9	4	8	4	1
1	2	3	4	9	5	8	3	1
1	2	3	4	5	9	8	2	1
1	2	3	4	5	8	9	1	1
							21	6

Aufgabe 3

Insertionsort:

8	3	7	6	2	4	Vergleiche	Verschiebungen
3	8	7	6	2	4	1	1
3	7	8	6	2	4	2	1
3	6	7	8	2	4	3	2
2	3	6	7	8	4	4	4
2	3	4	6	7	8	4	3
						14	11

Aufgabe 4

Insertionsort:

3	1	4	2	7	9	Vergleiche	Verschiebungen
1	3	4	2	7	9	1	1
1	3	4	2	7	9	1	0
1	2	3	4	7	9	3	2
1	2	3	4	7	9	1	0
1	2	3	4	7	9	1	0
						7	3

Aufgabe 5

Bubblesort:

3	1	4	2	7	9	Vergleiche	Vertauschungen
1	3	4	2	7	9	1	1
1	3	2	4	7	9	2	1
1	3	2	4	7	9	2	0
1	2	3	4	7	9	2	1
1	2	3	4	7	9	2	0
1	2	3	4	7	9	3	0
1	2	3	4	7	9	2	0
1	2	3	4	7	9	1	0
						15	3

Aufgabe 6

Bubblesort:

7	5	4	1	Vergleiche	Vertauschungen
5	7	4	1	1	1
5	4	7	1	1	1
5	4	1	7	1	1
4	5	1	7	1	1
4	1	5	7	1	1
1	4	5	7	1	1
				6	6

Aufgabe 7

Quicksort:

8	1	3	2	7	9	4	Vergleiche	Vertauschungen
8	1	3	2	7	9	4	1	0
1	8	3	2	7	9	4	1	1
1	3	8	2	7	9	4	1	1
1	3	2	8	7	9	4	1	1
1	3	2	8	7	9	4	1	0
1	3	2	8	7	9	4	1	0
1	3	2	4	7	9	8	0	1
1	3	2		7	9	8	1	1
1	3	2		7	9	8	1	0
1	2	3		7	9	8	0	1
1		3		7	9	8	0	1
		3		7	9	8	0	1
				7	9	8	1	1
				7	9	8	1	0
				7	8	9	0	1
				7		9	0	1
						9	0	1
							10	12

Aufgabe 8

Quicksort:

2	4	7	1	Vergleiche	Vertauschungen
2	4	7	1	1	0
2	4	7	1	1	0
2	4	7	1	1	0
1	4	7	2	0	1
	4	7	2	1	0
	4	7	2	1	0
	2	7	4	0	1
		7	4	1	0
		4	7	0	1
			7	0	1
				6	4

Aufgabe 9

Mergesort (iterativ):

7	3	8	5	9	2	4	1
3	7	5	8	2	9	1	4
3	5	7	8	1	2	4	9
1	2	3	4	5	7	8	9

Aufgabe 10

Mergesort (iterativ):

6	5	8	9	3	6	0	7	4	2
5	6	8	9	3	6	0	7	2	4
5	6	8	9	0	3	6	7	2	4
0	3	5	6	6	7	8	9	2	4
0	2	3	4	5	6	6	7	8	9

Aufgabe 11

Countingsort:

A=[3, 2, 1, 1, 0, 1, 3, 2]

B=[1, 3, 2, 2]

A'=[0, 1, 1, 1, 2, 2, 3, 3]

Aufgabe 12

Countingsort:

A=[2, 5, 3, 2, 2, 3, 2, 5, 1, 2]

B=[0, 1, 5, 2, 0, 2]

A'=[1, 2, 2, 2, 2, 2, 3, 3, 5, 5]

Aufgabe 13

Ist ein Pivot-Element an der richtigen Position, muss es nicht weiter sortiert werden.

- (a) Bei der minimalen Rekursionstiefe wird das Array in zwei Teilarrays zerlegt, deren Längen sich um höchstens 1 unterscheiden. Danach genügt es, das grössere der beiden Arrays zu untersuchen:

$$100 \rightarrow 50 + \text{Pivot} + 49$$

$$50 \rightarrow 25 + \text{Pivot} + 24$$

$$25 \rightarrow 12 + \text{Pivot} + 12$$

$$12 \rightarrow 6 + \text{Pivot} + 5$$

$$6 \rightarrow 3 + \text{Pivot} + 2$$

$$3 \rightarrow 1 + \text{Pivot} + 1$$

also 6 Rekursionsschritte

- (b) Die maximalen Rekursionstiefe entsteht, wenn das Array bereits sortiert ist. In diesem Fall wird ein Array der Länge n in ein leeres Array und in eines der Länge $n - 1$ zerlegt.

$$100 \rightarrow 0 + \text{Pivot} + 99$$

$$99 \rightarrow 0 + \text{Pivot} + 98$$

$$\dots \rightarrow \dots$$

$$2 \rightarrow 0 + \text{Pivot} + 1$$

also 99 Rekursionsschritte

Aufgabe 14

- (a) Randomized-Quicksort: Wähle vor jeder Partitionierung zufällig ein Element aus dem (Teil-)Array aus und vertausche es mit dem Element an der Pivotposition.
- (b) Wähle: vor jeder Partitionierung zufällig 3 Elemente aus dem (Teil-)Array aus, bestimme davon den Median und vertausche diesen mit dem Element an der Pivotposition.

Aufgabe 15

	Worst Case	Best Case
Mergesort	$O(n \log n)$	$O(n \log n)$
Selection Sort	$O(n^2)$	$O(n^2)$
Counting Sort	$O(n)$	$O(n)$
Insertion Sort	$O(n^2)$	$O(n)$
Quicksort	$O(n^2)$	$O(n \log n)$

Aufgabe 16

- Wenn die zu sortierenden Daten gut gemischt sind, sollte man Quicksort verwenden.
- Wenn die zu sortierenden Daten bereits (teilweise) sortiert sind und noch genügend Arbeitsspeicher für eine Kopie des Arrays vorhanden ist, sollte man Mergesort verwenden.

Aufgabe 17

Die Formel für den Basiswechsel bei Logarithmen

$$\log_b x = \frac{\log_c x}{\log_c b} = \frac{1}{\log_c b} \cdot \log_c x$$

besagt, dass sich Logarithmen zu verschiedenen Basen jeweils um einen konstanten Faktor unterscheiden. Da ein konstanter Faktor in der asymptotischen Betrachtungsweise der Laufzeitkomplexität ignoriert wird, können wir eine uns passende Basis für Logarithmen wählen.

$$T(10^6) = c \cdot 10^6 \log_{10} 10^6 = c \cdot 6 \cdot 10^6 = 3.6 \text{ s}$$

$$c = 0.6 \cdot 10^{-6} \text{ s}$$

$$T(10^7) = c \cdot 10^7 \log_{10} 10^7 = 0.6 \cdot 10^{-6} \text{ s} \cdot 7 \cdot 10^7 = 42 \text{ s}$$

Aufgabe 18

```
1 def isSorted(A):
2     for i in range(1, len(A)):
3         if A[i-1] > A[i]:
4             return False
5     return True
```