Für Array `A` der Länge `n`:

SelectionSort

```
for ende = n - 1 ... 1:
    max = 0
    for index = 1 ... ende:
        if A[index] > A[max]:
            max = index
    tausche A[max] mit A[ende]
```

BubbleSort

```
for ende = n - 2 ... 1:
    for index = 0 ... ende:
        if A[index] > A[index + 1]:
            tausche A[index] mit A[index + 1]
```

InsertionSort

```
for index = 1 ... n - 1:
    key = A[index]
    position = index
    while position > 0 and A[position - 1] > key:
        A[position] = A[position - 1]
        position = position - 1
    A[position] = key
```

CocktailSort

```
swapped = True
start = 0
ende = n - 1
while swapped:
    swapped = False
    for i = start ... ende:
        if A[i] > A[i + 1]:
            tausche A[i] mit A[i + 1]
            swapped = True
    if not swapped:
        break
    swapped = False
    ende = ende - 1
```

```
    for i = ende - 1 ... start - 1:
        if A[i] > A[i + 1]:
            tausche A[i] mit A[i+1]
            swapped = True
    start = start + 1
```

GnomeSort

```
index = 0
while index < n:
    if index == 0 or A[index] >= A[index - 1]:
        index = index + 1
    else:
        tausche A[index] mit A[index - 1]
        index = index - 1
```

HeapSort

```
def heapify(A, n, i):
    largest = i
    left = 2 * i + 1
    right = 2 * i + 2
    if left < n and A[left] > A[largest]:
        largest = left
    if right < n and A[right] > A[largest]:
        largest = right
    if largest != i:
        tausche A[i] mit A[largest]
        heapify(A, n, largest)

for start = n // 2 - 1 ... 0:
    heapify(A, n, start)
for ende = n - 1 ... 0:
    tausche A[0] mit A[ende]
    heapify(A, ende, 0)
```

MergeSort

```
def merge(A, left, mid, right):
    L = A[left ... mid]
    R = A[mid + 1 ... right]
    i = j = 0
    k = left
    while i < len(L) and j < len(R):
```

```
        if L[i] <= R[j]:
            A[k] = L[i]
            i = i + 1
        else:
            A[k] = R[j]
            j = j + 1
        k = k + 1
    while i < len(L):
        A[k] = L[i]
        i = i + 1
        k = k + 1
    while j < len(R):
        A[k] = R[j]
        j = j + 1
        k = k + 1

def merge_sort(A, left, right):
    if left < right:
        mid = (left + right) // 2
        merge_sort(A, left, mid)
        merge_sort(A, mid + 1, right)
        merge(A, left, mid, right)

merge_sort(A, 0, n - 1)
```

Quicksort

```
def partition(A, low, high):
    pivot = A[high]
    i = low - 1
    for j = low ... high - 1:
        if A[j] <= pivot:
            i = i + 1
            tausche A[i] mit A[j]
    tausche A[i + 1] mit A[high]
    return i + 1

def quick_sort(A, low, high):
    if low < high:
        pi = partition(A, low, high)
        quick_sort(A, low, pi - 1)
        quick_sort(A, pi + 1, high)

quick_sort(A, 0, n - 1)
```