Quicksort

```
QUICKSORT(A, p, r)
1  if p < r
2       q = PARTITION(A, p, r)
3       QUICKSORT(A, p, q − 1)
4       QUICKSORT(A, q + 1, r)

PARTITION(A, p, r)
1  x = A[r]
2  i = p − 1
3  for j = p to r − 1
4       if A[j] ≤ x
5            i = i + 1
6            tausche A[i] mit A[j]
7  tausche A[i + 1] mit A[r]
8  return i + 1
```

Mergesort

```
MERGESORT(A, p, r)
1  if p < r
2       q = ⌊(p + r)/2⌋
3       MERGESORT(A, p, q)
4       MERGESORT(A, q + 1, r)
5       MERGE(A, p, q, r)

MERGE(A, p, q, r)
1   n_1 = q − p + 1
2   n_2 = r − q
3   seien L[1..n_1 + 1] und R[1..n_2 + 1] neue Arrays
4   for i = 1 to n_1
5        L[i] = A[p + i − 1]
6   for j = 1 to n_2
7        R[j] = A[q + j]
8   L[n_1 + 1] = ∞
9   R[n_1 + 1] = ∞
10  i = 1
11  j = 1
12  for k = p to r
13       if L[i] ≤ R[j]
14            A[k] = L[i]
15            i = i + 1
16       else A[k] = R[j]
17            j = j + 1
```

Countingsort

Input: $A[1..n]$

```
COUNTINGSORT(A, k)
 1  n = A.length
 2  sei B[1..n] ein neues Array
 3  sei C[0..k] ein neues Array
 4  for i = 0 to k
 5      C[i] = 0
 6  for j = 1 to n
 7      C[A[j]] = C[A[j]] + 1
 8  for i = 1 to k
 9      C[i] = C[i] + C[i − 1]
10  for j = n downto 1
11      B[C[A[j]]] = A[j]
12      C[A[j]] = C[A[j]] − 1
```