Algorithm 1: CountInversions.

Input: Array $A[0 \dots n-1]$.
Output: The number of inversions in A.

if $n = 1$ **then**
    return 0;
else
    copy $A[0 \dots \lfloor n/2 \rfloor - 1]$ to $B[0 \dots \lfloor n/2 \rfloor - 1]$;
    copy $A[\lfloor n/2 \rfloor \dots n-1]$ to $C[0 \dots \lceil n/2 \rceil - 1]$;
    $il \leftarrow$ **CountInversions**(B);
    $ir \leftarrow$ **CountInversions**(C);
    $im \leftarrow$ **Merge**(B, C, A);
    return $il + ir + im$;

Algorithm 2: Merge.

Input: Two sorted arrays $B[0 \dots p-1]$ and $C[0 \dots q-1]$, and an array $A[0..p+q-1]$.
Output: The number of inversions involving an element from B and an element from C.
Modifies: A.

$count \leftarrow 0$;
$i \leftarrow 0$; $j \leftarrow 0$; $k \leftarrow 0$;
while $i < p$ **and** $j < q$ **do**
1   **if** ... **then**
        $A[k] \leftarrow B[i]$; $i \leftarrow i + 1$;
    else
        $A[k] \leftarrow C[j]$; $j \leftarrow j + 1$;
2         $count \leftarrow count + ...$;
    $k \leftarrow k + 1$;
if $i = p$ **then**
    copy $C[j \dots q-1]$ to $A[k \dots p+q-1]$;
else
    copy $B[i \dots p-1]$ to $A[k \dots p+q-1]$;
return $count$;

- Line 1: $B[i] \leq C[j]$
- Line 2: $p - i$