## Warm-up

**Problem 1.** Sort the following array using merge-sort: $A = [5, 8, 2, 0, 23, 786, -2, 65]$. Give all arrays on which recursive calls are made and show how they are merged back together.

**Problem 2.** Consider the following algorithm.

```
function REVERSE(A)
    if |A| = 1 then
        return A
    else
        B ← first half of A
        C ← second half of A
        return concatenate REVERSE(C) with REVERSE(B)
```

Let $T(n)$ be the running time of the algorithm on an instance of size $n$. Write down the recurrence relation for $T(n)$ and solve it by unrolling it.

## Problem solving

**Problem 3.** Given an array $A$ holding $n$ objects, we want to test whether there is a *majority* element; that is, we want to know whether there is an object that appears in more than $n/2$ positions of $A$.

Assume we can test equality of two objects in $O(1)$ time, but we cannot use a dictionary indexed by the objects. Your task is to design an $O(n \log n)$ time algorithm for solving the majority problem.

a) Show that if $x$ is a majority element in the array then $x$ is a majority element in the first half of the array or the second half of the array

b) Show how to check in $O(n)$ time if a candidate element $x$ is indeed a majority element.

c) Put these observation together to design a divide an conquer algorithm whose running time obeys the recurrence $T(n) = 2T(n/2) + O(n)$

d) Solve the recurrence by unrolling it.

**Problem 4.** Let $A$ be an array with $n$ distinct numbers. We say that two indices $0 \leq i < j < n$ form an inversion if $A[i] > A[j]$. Modify merge sort so that it computes the number of inversions of $A$.

**Problem 5.** Given a sorted array $A$ containing distinct non-negative integers, find the smallest non-negative integer that isn't stored in the array. For simplicity, you can assume there is such an integer, i.e., $A[n-1] > n-1$

Example: $A = [0, 1, 3, 5, 7]$, result: 2

**Problem 6.** Design an $O(n)$ time algorithm for the majority problem.