

# Laboration 2

Due 2022-12-01

For each algorithm designed, you should

- give a complete and unambiguous high-level description (step-wise) of your algorithm in plain English/Swedish; and
  - implement your algorithm (using either Python or Java).
1. Given  $n$  ( $n \geq 3$ ) distinct elements, design two algorithms to compute the first three smallest elements using an incremental and a divide-and-conquer approach, respectively. Both your algorithms should return a triple  $(x, y, z)$  such that  $x < y < z <$  (the rest  $n - 3$  input elements) and run in linear time in the worst case. Show that your algorithms are correct and calculate the *exact* number of comparisons used by the algorithms. You may assume that  $n = 3 \times 2^{k-1}$  for some positive integer  $k$ . *Hint: One can use the induction technique to show the correctness. Check Chapter 4 for more examples of performance analyses.*
  2. Given an array  $A = \langle a_1, a_2, \dots, a_n \rangle$  of non-zero real numbers, the problem is to find a subarray  $\langle a_i, a_{i+1}, \dots, a_j \rangle$  (of consecutive elements) such that the sum of all the numbers in this subarray is maximum over all possible consecutive subarrays. Design a divide and conquer algorithm to compute such a maximum sum. You do not need to actually output such a subarray; only returning the maximum sum. Write only *one* recursive function to implement your algorithm. Built-in functions or methods for strings or lists must not be used. Your algorithm should run in  $O(n)$  time in the worst case. You may assume that  $n = 2^k$  for some positive integer  $k$ .

## Report

Each group must hand in one report for each part. The report can be written in either Swedish or English and should not be handwritten.

*Before submitting your report, you should discuss your solution to the laboration (design, implementation, and report) with your lab-assistant.*