# Informatics II, Spring 2023, Exercise 4

Publication of exercise: March 13, 2023
Publication of solution: March 19, 2023
Exercise classes: March 20 - March 24, 2023

Note, in this exercise $\log n$ refers to log base 2, ($\log_2 n$), if not stated otherwise.

## Task 1: Divide and Conquer

The maximum-subarray algorithm finds the contiguous subarray that has the largest sum within an **unsorted** array $A$ with $n$ integers. For example, for array `A = [-2, -5, 6, -2, -3, 1, 5]`, the maximum subarray is `[6, -2, -3, 1, 5]`.
The algorithm works as follows:
Firstly, it divides the input array into two equal partitions: **I** (A[0]...A[*mid*]) and **II** (A[*mid*+1]...A[n-1]). Afterwards, it calls itself recursively on both partitions to find the maximum subarray of each partition. The combination step decides the maximum-subarray by comparing three arrays: the maximum-subarray from the left part, the maximum-subarray from the right part, and the maximum-subarray that overlaps the middle. The maximum-subarray that overlaps the middle is determined by considering all elements to the left and all elements to the right of the middle.

**1.1** Based on the above algorithm description, draw a tree that illustrates the process of determining the maximum subarray in array `A = [-1, 2, -4, 1, 9, -6, 7, -3, 5]`.

**1.2** Provide a C code for maximum-subarray algorithm.

**1.3** What is the recurrence relation of the algorithm and its asymptotic tight bound.

## Recurrences

### Task 2: Substitution Method

Consider the recurrence $T_a$ for the following questions:

$$T_a(n) = \begin{cases} 1, & \text{if } n = 1 \\ 2T(n-1) + c_1, & \text{otherwise} \end{cases} \tag{1}$$

**2.1** In order to find an upper bound for the recurrence $T_a(n)$ with the substitution method, you need to make an educated guess what a good upper bound could be. To make it easier for you to make a good guess, write a short C program that computes the values for $T(n)$ and that prints out the results after each recursive step. What function of $n$ could result in the values that you observe? Vary $n$ and $c_1$ to get a better intuition.

**2.2** Now use the substitution method to find an upper bound for $T_a(n)$. Use the guessed upper bound from above.

## Task 3: Repeated Backward Substitution

Consider the recurrence $T_b$ for the following questions, defined as:

$$T_b(n) = \begin{cases} 1, & \text{if } n = 1 \\ 2T(n/2) + n, & \text{otherwise} \end{cases}$$

Use the repeated backward substitution method to find an upper bound for $T_b(n)$.

## Task 4: Recursion Tree Method

Consider the following Algorithm for the subsequent questions.

**Algo:** RecursiveAlgo(n)

```
if n ≤ 1 then
    return
int t = 0;
for i=0 to max(n,1) do
    t = t + i ;
RecursiveAlgo(n/10) ;
RecursiveAlgo(9n/10) ;
```

**4.1** Analyse the algorithm above and find the recurrence relation $T(n)$ that determines its runtime. What is the base case?

**4.2** Use the recursion tree method to find an upper bound for the runtime of the algorithm.

## Task 5: Master Method

Use the Master Method to calculate the asymptotic tight bound for the following recurrences. Write down which case applies, as well as $a$, $b$ and $f(n)$.

**5.1** $T(n) = 2T(\frac{n}{4}) + \sqrt{n} + 5$

**5.2** $T(n) = 12T(\frac{n}{8}) + n^3$