Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik, Anja Rey

Exercise Sheet 2

for the lecture on

Advanced Programming and Algorithms

Submission until **Monday, 30th October, 12:30 pm**.
Discussion in the exercise classes on 6th, 9th, and 10th November, 2023.

Problem 1 to hand in: *Running Time*

The following pseudocode describes the `bubble_sort` algorithm. This is another way to solve the SORTING computation problem: Given a finite sequence A of pairwise distinct integers (and its length), return A sorted ascendingly.

```
bubble_sort($A[0..n-1], n$):
for $j \leftarrow n-1$ down to $1$ do
    for $i \leftarrow 0$ to $j-1$ do
        if $A[i] > A[i+1]$ then
            key $= A[i]$
            $A[i] = A[i+1]$
            $A[i+1] =$ key
return $A$
```

a) How does `bubble_sort` work in comparison to `insertion_sort`? Describe it intuitively in one or two sentences.

b) Analyse the asymptotic worst-case running time of `bubble_sort`:

- For each line of code, write down the number of running steps (dependent on the input size) in the worst case.
- Sum up the total number $T(n)$ of steps the algorithm needs in the worst case for an input of size n.
- Provide a function $f(n)$ as a representative upper bound in $\mathcal{O}$-notation.
- Proof formally that $T(n) \in \mathcal{O}(f(n))$ holds.

c) What is the (asymptotic) average running time? Briefly argue why.

Problem 2 as a programming exercise: *Refactoring*

Implement `bubble_sort` in Python.

How can you optimise the readability and reusability of your code with respect to the following criteria?

- consistency (e.g. spacing)
- expressivity (e.g. variable names)
- function extendability (unique purpose, brevity, testability)
- avoiding redundancy

Discuss: How do these changes affect the (best-case / worst-case / average) running time of the algorithm?

Problem 3 for discussion: *Space Complexity*

Similar to the running time of an algorithm, the memory space required by an algorithm can be analysed theoretically.

a) Define a formal notion of space complexity.

b) Analyse the space complexity of `insertion_sort`

Problem 4 for discussion: *Correctness*

Consider the following algorithm.

`do_something`$(A[0..n-1], n)$:

```
1 $s \leftarrow 0$
2 for $i \leftarrow 1$ to $n$ do
3     $s \leftarrow s + 2 \cdot (i - 1) + 1$
4 return $s$
```

a) What happens here? State the computation problem solved by this algorithm.

b) State a loop invariant that holds at the beginning of each iteration of the for loop (lines 2 to 4).

c) Proof the loop invariant.

d) Use the loop invariant to show that indeed the algorithm solves the computation problem from subtask a).