University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn

Algorithms and Data Structures
Winter Term 2021/2022
Exercise Sheet 2

Exercise 1: $\mathcal{O}$-notation

Prove or disprove the following statements. Use the *set definition* of the $\mathcal{O}$-notation (lecture slides week 2, slides 11 and 12).

(a) $4n^3 + 8n^2 + n \in \mathcal{O}(n^3)$

(b) $2^n \in o(n!)$

(c) $2\log n \in \Omega((\log n)^2)$

(d) $\max\{f(n), g(n)\} \in \Theta(f(n) + g(n))$ for non-negative functions f and g.

Exercise 2: Sorting by asymptotic growth

Sort the following functions by their asymptotic growth. Write $g <_{\mathcal{O}} f$ if $g \in \mathcal{O}(f)$ *and* $f \notin \mathcal{O}(g)$. Write $g =_{\mathcal{O}} f$ if $f \in \mathcal{O}(g)$ *and* $g \in \mathcal{O}(f)$ (no proof needed).

$\sqrt{n}$	2^n	$n!$	$\log(n^3)$
3^n	n^{100}	$\log(\sqrt{n})$	$(\log n)^2$
$\log n$	$10^{100}n$	$(n+1)!$	$n \log n$
$2^{(n^2)}$	n^n	$\sqrt{\log n}$	$(2^n)^2$

Exercise 3: Stable Sorting

A sorting algorithm is called stable if elements with the same key remain in the same order. E.g., assume you want to sort the following strings where the sorting key is *the first letter by alphabetic order*:

$$[\text{“}tuv\text{”}, \text{“}adr\text{”}, \text{“}bbc\text{”}, \text{“}tag\text{”}, \text{“}taa\text{”}, \text{“}abc\text{”}, \text{“}sru\text{”}, \text{“}bcb\text{”}]$$

A *stable* sorting algorithm must generate the following output:

$$[\text{“}adr\text{”}, \text{“}abc\text{”}, \text{“}bbc\text{”}, \text{“}bcb\text{”}, \text{“}sru\text{”}, \text{“}tuv\text{”}, \text{“}tag\text{”}, \text{“}taa\text{”}]$$

A sorting algorithm is not stable (with respect to the sorting key) if it outputs, e.g., the following:

$$[\text{“}abc\text{”}, \text{“}adr\text{”}, \text{“}bbc\text{”}, \text{“}bcb\text{”}, \text{“}sru\text{”}, \text{“}taa\text{”}, \text{“}tag\text{”}, \text{“}tuv\text{”}]$$

(a) Which sorting algorithms from the lecture (except `CountingSort`) are *not* stable? Prove your statement by giving an appropriate example.

(b) Describe a method to make any sorting algorithm stable, without changing the *asymptotic* runtime. Explain.

Exercise 4: Running time

Give an asymptotically tight upper bound for the running time of the following algorithm as a function of n.

$s \leftarrow 0$
for $i = 1$ to n **do**
    $j = 1$
    while $j < i$ **do**
        $s \leftarrow s + i \cdot j$
        $j \leftarrow 2 \cdot j$
