University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
M. Ahmadi, P. Schneider

Algorithms Theory
Exercise Sheet 1

Due: Monday, 5th of November, 2018, 14:15 pm

Exercise 1: O-Notation *(3+4+5 Points)*

For a function $f(n)$, the set $\mathrm{O}\big(f(n)\big)$ contains all functions $g(n)$ that are *asymptotically* not growing faster than $f(n)$. The set $\Omega\big(f(n)\big)$ contains all functions $g(n)$ with $f(n) \in \mathrm{O}\big(g(n)\big)$. Finally, $\Theta\big(f(n)\big)$ contains all functions $g(n)$ for which $f(n) \in \mathrm{O}\big(g(n)\big)$ *and* $g(n) \in \mathrm{O}\big(f(n)\big)$. This is formalized as follows:

$$\mathrm{O}\big(f(n)\big) := \{g(n) \mid \exists c > 0, n_0 \in \mathbb{N},\ \forall n \geq n_0 : g(n) \leq cf(n)\}$$
$$\Omega\big(f(n)\big) := \{g(n) \mid \exists c > 0, n_0 \in \mathbb{N}\ \forall n \geq n_0 : g(n) \geq cf(n)\}$$
$$\Theta\big(f(n)\big) := \{g(n) \mid \exists c_1, c_2 > 0, n_0 \in \mathbb{N}\ \forall n \geq n_0 : c_1 f(n) \leq g(n) \leq c_2 f(n)\}$$

State whether the following claims are correct or not. Prove or disprove with the definitions above.

(a) $n! \in \Omega\big(n^2\big)$

(b) $\sqrt{n^3} \in \mathrm{O}(n \log n)$ ***Hint:*** *For all* $\varepsilon > 0$ *there is an* $n_0 \in \mathbb{N}$ *such that for all* $n \geq n_0 :\ \log_2 n \leq n^{\varepsilon}$.

(c) $2^{\sqrt{\log_2 n}} \in \Theta(n)$

Exercise 2: Sort Functions by Asymptotic Growth *(5 Points)*

Use the definition of the O-notation to give a sequence of the functions below, which is ordered by asymptotic growth (ascending). Between two consecutive elements g and f in your sequence, insert either $\prec$ (in case $g \in \mathrm{O}(f)$ *and* $f \notin \mathrm{O}(g)$) or $\simeq$ (in case $g \in \mathrm{O}(f)$ *and* $f \in \mathrm{O}(g)$).

Note: *No formal proofs required, but you loose* $\frac{1}{2}$ *point for each error.*

n^2	$\sqrt{n}$	$2^{\sqrt{n}}$	$\log(n^2)$
$2^{\sqrt{\log_2 n}}$	$\log(n!)$	$\log(\sqrt{n})$	$(\log n)^2$
$\log n$	$10^{100}n$	$n!$	$n \log n$
$2^n/n$	n^n	$\sqrt{\log n}$	n

Exercise 3: Master Theorem for Recurrences *(5 Points)*

Use the *Master Theorem* for recurrences, to fill the following table. That is, in each cell write $\Theta\big(g(n)\big)$, such that $T(n) \in \Theta\big(g(n)\big)$ for the given parameters $a, b, f(n)$. Assume $T(1) \in \Theta(1)$. Additionally, in each cell note the case you used (1st, 2nd or 3rd by the order given in the lecture). We filled out one cell as an example.

Note: *You loose* $\frac{1}{2}$ *point if the complexity class is wrong and another* $\frac{1}{2}$ *if the case is wrong.*

$T(n) = aT(\frac{n}{b}) + f(n)$	$a = 16, b = 2$	$a = 1, b = 2$	$a = b = 3$
$f(n) = 1$	$\Theta(n^4)$, 1st		
$f(n) = n^3$			
$f(n) = n^4 \log n$			

Exercise 4: Peak Element *(5+4 Points)*

You are given an array $A[1 \dots n]$ of n integers and the goal is to find a peak element, which is defined as an element in A that is equal to or bigger than its direct neighbors in the array. Formally, $A[i]$ is a peak element if $A[i-1] \leq A[i] \geq A[i+1]$. To simplify the definition of peak elements on the rims of A, we introduce *sentinal-elements* $A[0] = A[n+1] = -\infty$.

(a) Give an algorithm with runtime $O(\log n)$ (measured in the number of read operations on the array) which returns the position i of a peak element.

(b) Prove that your algorithm always returns a peak element, give a recurrence relation for the runtime and use it to prove the runtime.

Exercise 5: Frequent Numbers *(5+4 Points)*

You are given an Array $A[0 \dots n{-}1]$ of n integers and the goal is to determine frequent numbers which occur at least $n/3$ times in A. There can be at most three such numbers, if any exist at all.

(a) Give an algorithm with runtime $O(n \log n)$ (measured in number of array entries that are read) based on the divide and conquer principle that outputs the frequent numbers (if any exist).

(b) Argue why your algorithm is correct, give a recurrence relation for the runtime and use it to prove the runtime.