Albert-Ludwigs-Universität, Institut für Informatik
Prof. Dr. Fabian Kuhn
M. Ahmadi, P. Schneider

October 23, 2017

Algorithm Theory - Winter Term 2017/2018
Exercise Sheet 1

Hand in by Thursday 10:15, November 2, 2017

Exercise 1: Landau Notation - Formal Proofs (2+2+3+3 Points)

For a function $f(n)$, the set $\mathcal{O}\big(f(n)\big)$ contains all functions $g(n)$ that are *asymptotically* not growing faster than $f(n)$. This is formalized as follows:

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

The set $\Omega\big(f(n)\big)$ contains all functions $g(n)$ with $f(n) \in \mathcal{O}\big(g(n)\big)$. Finally, $\Theta\big(f(n)\big)$, contains all functions $g(n)$ for which both $f \in \mathcal{O}\big(g(n)\big)$ and $g \in \mathcal{O}\big(f(n)\big)$ is true. For each pair of functions from (a) to (c) **prove** whether $g(n) \in \mathcal{O}\big(f(n)\big)$ or $g(n) \in \Omega\big(f(n)\big)$ or both, i.e., $g(n) \in \Theta\big(f(n)\big)$.

Note: *You do not have to prove negative results* $g(n) \notin \mathcal{O}\big(f(n)\big)$, *it suffices to claim these correctly.*

(a) $g(n) = 100n,\ f(n) = 0.1 \cdot n^2$

(b) $g(n) = \sqrt[3]{n^2},\ f(n) = \sqrt{n}$

(c) $g(n) = \log_2(2^n \cdot n^3),\ f(n) = n$ ***Hint:*** *You may use that* $\log_2 n \leq n$ *for all* $n \in \mathbb{N}$.

(d) Fill out the following table (with ✓or ✗) based on whether the statement is true or false.

Note: *You loose* $\frac{1}{2}$ *points for each wrong cell.*

	$g \in \mathcal{O}(f)$	$f \in \mathcal{O}(g)$	$g \in \Omega(f)$	$f \in \Omega(g)$	$g \in \Theta(f)$	$f \in \Theta(g)$
a)						
b)						
c)						

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

Using the $\mathcal{O}$-notation definition, give an ordered sequence of the following functions based on their asymptotic growth . Between each consecutive elements g and f in your list, insert either $\prec$ (in case $g \in \mathcal{O}(f)$ *and* $f \notin \mathcal{O}(g)$) or $\simeq$ (in case $g \in \mathcal{O}(f)$ *and* $f \in \mathcal{O}(g)$).

Note: *You loose* $\frac{1}{2}$ *points for each error.*

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

Exercise 3: Recurrence Relations (3+3+3 Points)

Let α, β be *constants* and let $T(n)$ be a *monotonously increasing* function in $n \in \mathbb{N}$ with

$$T(1) \leq \alpha, \quad T(n) \leq 4 \cdot T(n/4) + \beta \cdot n.$$

(a) Guess an upper bound for $T(n)$ (as tight as possible, with the given knowledge about $T(n)$, e.g. by repeated substitution, cf. lecture). You may assume that $n = 4^k$ for some $k \in \mathbb{N}$.

(b) Prove your upper bound via induction on k. Assume that $n = 4^k$ for some $k \in \mathbb{N}$.

(c) Give an *asymptotic* upper bound ($\mathcal{O}$-Notation) for $T(n)$, assuming $n = 4^k$ for some $k \in \mathbb{N}$. Prove that the same bound applies to $T(n)$ for *all* $n \in \mathbb{N}$ (i.e., if n is not necessarily a power of 4).

Exercise 4: 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). We filled out one cell as an example.

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

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

Exercise 5: Almost Closest Pairs (4+7 Points)

In the lecture, we discussed an $\mathcal{O}(n \log n)$-time divide-and-conquer algorithm to determine the closest pair of points among a set of n points $\{x_1, \ldots, x_n\} \in \mathbb{R}^2$ on the real plane. Assume that we are not only interested in the closest pair of points, but in all pairs of points that are at distance at most twice the distance between the closest two points.

(a) How many such pairs of points can there be? It suffices to give your answer using the $\mathcal{O}$-notation.

(b) Devise an algorithm that outputs a list with all pairs of points at distance at most twice the distance between the closest two points. Describe what you have to change compared to the closest pair algorithm of the lecture and analyze the running time of your algorithm.