Artificial Intelligence for Robotics – Lab

Prof. Dr. Erwin A. Prassler: *erwin.prassler@h-brs.de*

Maximilian Schöbel: *maximilian.schoebel@h-brs.de*

Patrick Nagel: *patrick.nagel@h-brs.de*

Assignment 2

Due Date: Tuesday, 23.10.2018, 08:00

1. *(1 Points)* Why is a special notation needed to classify algorithms? Doesn't it suffice to merely measure the runtime in seconds? Explain.

2. *(2 Points)* Let $g : \mathbb{N} \to \mathbb{R}_+$ be an arbitrary function. The set of functions $f : \mathbb{N} \to \mathbb{R}_+$, which do not grow faster than the function g after a specific point n_0, is denoted as $O(g(n))$. More specifically: $O(g(n)) := \{f(n) \mid \exists\ c \in \mathbb{R}$ and $\exists\ n_0 \in \mathbb{N} : 0 \leq f(n) \leq c \cdot g(n)\ \forall\ n \geq n_0\}$ Formally proof:

   - $f(n) = 100n^2 \in O(n^2)$
   - $f(n) = n^6 + 100n^5 \in O(n^6)$

3. *(1 Points)* What is the running time of the following python-code in O-Notation? Assume, that `ANY_CONST` is an arbitrary constant in your program, which receives a 2d array `arr` as parameter.

```
sum = 0
for i in range(0, J):
    for j in range(0, K):
            if arr[i][j] <= ANY_CONST:
                sum = sum + arr[i][j]
print(sum)
```

4. *(3 Points)* For each function $f(n)$ and time t in the following table, determine the largest size n of a problem that can be solved in time t, assuming that the algorithm to solve the problem takes $f(n)$ microseconds.

	1 second	1 minute	1 hour	1 day	1 month	1 year	1 century
$\lg n$							
$\sqrt{n}$							
n							
$n \lg n$							
n^2							
n^3							
2^n							
$n!$							