# MAS AI Assignment 2

Simon Deussen

November 15, 2020

## Task 1

In the first part of my answer I will explain why measuring the runtime in seconds does not work, and in the second part I am going to explain the notation.

We want to compare the runtime of algorithms, for this it is important that the classification works *regardless of the computer running it.* This requirement immediately dismisses the measuring in seconds. How would it be possible to map the runtime in seconds between different computer with different instruction sets and clock speeds?

Secondly, an algorithm works on a given input. If we only look at the runtime, we can not possibly know how any algorithm works with different amounts of input data. Because of those reasons, we can see that working with absolute runtime in seconds is only getting complicated and wont allow us important comparisons.

The special *Big o notation* is a measure that describes how many steps an algorithm has to perform in relation to the amount of input data. By looking at abstract steps, this notation removes many nasty details from the discussion of algorithmic runtime. It works in a way that it describes to growth of needed steps in relation to number of input data. Which allows an easy comparisons of algorithms.

## Task 2

Prove the following:

- $f(n) = 100n^2 \in O(n^2)$
  is True because:
  $f(n) \leq cg(n^2) \forall c > 100$
- $f(n) = n^6 + 100n^5 \in O(n^6)$
  is True because:
  $f(n) \leq cg(n^6) \forall c > 100$

because

$cn^6 > 100n^5 \forall c > 100$

# Task 3

First, lets annotate the given code with the cost per line and execution times:

| code | cost | times |
|---|---|---|
| `sum = 0` | $c_0$ | 1b |
| `for i in range(0, J):` | $c_1$ | $J$ |
| `  for j in range(0, K):` | $c_2$ | $J * K$ |
| `    if arr[i][j] <= ANY_CONST:` | $c_3$ | $n_0 \leq J * K$ |
| `      sum = sim + arr[i][j]` | $c_4$ | $n_1 \leq J * K$ |
| | | |
| `print(sum)` | $c_5$ | 1 |

We see that the total cost equals: $c_0 + c_1 J + c_2 JK + c_3 n_0 + c_4 n_1 + c_5 \forall n_0, n_1 \leq JK$ For the *Big O notation* only the biggest term is relevant: $c_2 JK$ Further, the constant is irrelevant and we assume the worst case, $J = K$ which leaves: $J^2$. Because there is a $g(n) = cn^2 > J^2 \forall c > 0$ we can say, this code has the complexity of $O(n^2)$.

# Task 4

For this task we have to calculate all the entries of given table. Where i could invert the given function

| | 1 second | 1 minute | 1 hour | 1 day | 1 month | 1 year | 1 century |
|---|---|---|---|---|---|---|---|
| $lg(n)$ | $10^{60}$ | $10^{120}$ | $10^{180}$ | $10^{204}$ | $10^{234}$ | $10^{246}$ | $10^{256}$ |
| $\sqrt{n}$ | $10^{12}$ | $3.6 * 10^{15}$ | $1.4 * 10^{19}$ | $7.5 * 10^{21}$ | $6.7 * 10^{24}$ | $9.7 * 10^{26}$ | $9.7 * 10^{28}$ |
| $n$ | $10^6$ | $6 * 10^7$ | $3.6 * 10^9$ | $8.6 * 10^{10}$ | $2.6 * 10^{12}$ | $3.1 * 10^{13}$ | $3.1 * 10^{14}$ |
| $nlg(n)$ | | | | | | | |
| $n^2$ | $10^3$ | $7.7 * 10^3$ | $6.0 * 10^4$ | $2.9 * 10^5$ | $1.6 * 10^6$ | $5.6 * 10^5$ | $1.7 * 10^7$ |
| $n^3$ | $10^2$ | $3.9 * 10^2$ | $1.5 * 10^3$ | $4.4 * 10^3$ | $1.3 * 10^4$ | $3.14 * 10^4$ | $6.8 * 10^4$ |
| $2^n$ | 19.9 | 25.8 | 31.7 | 36.3 | 41.2 | 44.8 | 48.1 |
| $n!$ | 9 | 10 | 12 | 13 | 15 | 16 | 16 |

$nlg(n)$ I do not know how to calculate this efficiently.

$2^n$ The inverse is: $log_2(x)$

$n!$ I made an approximation (brute force) using a python script.