Computational Complexity

Exercise Session 1

Definition 1. Take two functions $f, g : \mathbb{N} \to \mathbb{N}$.

$$
\begin{aligned}
f \text{ is } O(g) &\quad \text{iff there exist } c, n_0 \in \mathbb{N} \text{ such that } f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0 && (\leq)\\
f \text{ is } o(g) &\quad \text{iff for every } \epsilon > 0 \text{ there exists } n_0 \in \mathbb{N} \text{ such that } f(n) \leq \epsilon \cdot g(n) \text{ for all } n \geq n_0 && (<)\\
&\quad \text{or equivalently, iff} \quad \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 &&\\
f \text{ is } \Omega(g) &\quad \text{iff} \quad g \text{ is } O(f) && (\geq)\\
f \text{ is } \omega(g) &\quad \text{iff} \quad g \text{ is } o(f) && (>)\\
f \text{ is } \Theta(g) &\quad \text{iff} \quad f \text{ is } O(g) \text{ and } f \text{ is } \Omega(g) && (=)
\end{aligned}
$$

Exercise 1.

1. Show that $(\log n)$ is $o(n)$.[1]
   - *Hint:* show by induction that for arbitrary d it holds that: $d \log n \leq n$ for sufficiently large n. (You may use the fact that for each d it holds that $n + 1 < 2^{1/d} \cdot n$ for sufficiently large n.)
   - *Hint:* alternatively, use the limit characterization, and use L'Hôpital's Rule.[2]
2. Show that the constant function $f(n) = 1$ is $o(n)$.
3. Show that n is $o(n^2)$.
4. Show that n^d is $O(2^n)$ for each $d \in \mathbb{N}$.
   - *Hint:* use the fact that for arbitrary d it holds that: $d \log n \leq n$ for sufficiently large n.
5. Show that $n/2$ is $\Theta(n)$.

Exercise 2 (Inspired by exercise 1.14(b) from the book (Arora & Barak, 2009))**.**
In this exercise you will work with various levels of description for algorithms. For the remainder of the course, you may use any level of description for algorithms—whichever is most convenient in the context—as long as the details that you are omitting can be filled in straightforwardly.[3]

1. An undirected graph $G = (V, E)$ consists of a finite set V of vertices and a finite set $E \subseteq \binom{V}{2}$ of edges (subsets of V of size exactly 2). Describe how one can describe any graph as a string using the alphabet $\Sigma_{\text{graphs}} = \{\texttt{(}, \texttt{)}, \texttt{,}, \texttt{0}, \dots, \texttt{9}\}$. The symbols in Σ are written in typewriter font to differentiate them from symbols in the meta-language.
   - The names of vertices do not matter, so you may give each vertex an arbitrary name in your encoding.
   - *Hint:* give the vertices an arbitrary numbering and represent each edge as a pair of numbers. (*Note:* there are multiple possible suitable representations—so yours may differ from this suggestion.)
2. Describe an algorithm that checks whether a given string over the alphabet Σ_{graphs} is a valid representation of a graph G. Does this algorithm run in polynomial time? If so, explain why, and if not, explain why not.
3. Give a high-level description of an algorithm A that computes whether a given undirected graph G contains a triangle—that is, a triple of vertices v_1, v_2, v_3 such that $\{v_1, v_2\} \in E$, $\{v_1, v_3\} \in E$ and $\{v_2, v_3\} \in E$.
   Use descriptions at the level of detail of *"iterate over all vertices v_1," "check if $\{v_1, v_2\} \in E$," "if such-and-such-easy-to-check-condition is the case, return 1"* and similar statements.

[1]As is typical in computer science, log refers to the binary logarithm ($\log_2$).
[2]See, e.g., `https://en.wikipedia.org/wiki/L%27H%C3%B4pital%27s_rule`
[3]This can be a delicate balance, and unfortunately there is no clear-cut recipe for this, so use your own judgment.

4. Describe your algorithm A at a somewhat more detailed level of description.
   Use descriptions at the level of detail of *"let x_1 be a variable, and set $x_1 := 1$," "read the largest number in the input string, and set x_2 to this value,"*, *"while $x_1 < x_2$, do ...," "if (x_1,x_2) occurs in the input, do ...,"* and similar statements. In other words, use something like pseudocode.[4]

5. If you were to spell out your algorithm A in full detail as a Turing machine, how would you approach this? For example, how many tapes would you use? What would you intend to write on these tapes? Roughly, what groups of states $q \in Q$ would you create, and for what purpose?

   - *Note:* spelling out the Turing machine corresponding to A is tedious (and typically little fun). So please don't go into full detail, but reflect on what you would have to do if you *were to.*

6. Does your algorithm A run in polynomial time? If so, explain why, and if not, explain why not.

[4] See, e.g., `https://en.wikipedia.org/wiki/Pseudocode`