Universität Leipzig
Institute of Computer Science
Winter term 2026/2027

Andreas Maletti

Exercises for the lecture

Foundations of Complexity Theory

Solutions to Exercise Sheet 3

for the exercise on November 4, 2026

Exercise 3.1

Formulate the theorem of Borodin-Trakhtenbrot (page 8 of 2nd lecture) for DSPACE.
Does it also hold for NTIME resp. NSPACE?

Solution: It holds in all mentioned variants. For DSPACE we have:

Let $f\colon \mathbb{N} \to \mathbb{N}$ be a total, computable, and monotonic increasing function with $f(n) \geq n$ for all $n \in \mathbb{N}$. There effectively exists a total, computable, and monotonic increasing function $g\colon \mathbb{N} \to \mathbb{N}$ with $g(n) \geq \log n$ for all $n \in \mathbb{N}$ and $\mathrm{DSPACE}(g) = \mathrm{DSPACE}(g\,; f)$.

In the proof we take the supremum over $\max\{|\kappa_0|, \ldots, |\kappa_n|\}$ instead of n in the definition of $t_\ell(m)$. To compute $g(m)$ we let $s \leftarrow \max(\log m, g(m-1))$. It remains to settle how to test $t_\ell(m) \notin [s, f(s)]$ effectively in this case. As seen in Lemma 1.7, the number of $f(s)$-space-bounded configurations is finite. If we reach more than $a \cdot n \cdot b^{f(s)}$ configurations from I_w, then TM obviously has space consumption larger than $f(s)$. If fewer configurations are reachable, then it is finite and closed under successors, so we can determine the exact space consumption. Since the results are known for the finitely many inputs w with $|w| < \ell - 2$, we need no space for these inputs. For the remaining inputs at most space $g(m)$ is needed, which proves the result.

This adjusted proof also works in the nondeterministic case and the adjustment for NTIME is straightforward since a repetition of configurations for a given input also yields a loop and unbounded time complexity for that input.

Exercise 3.2

Remember the theorem on linear tape compression (page 67 of 2nd lecture):

$$\mathrm{DSPACE}(f) \subseteq \mathrm{DSPACE}(g) \text{ for all } \epsilon > 0 \text{ and } g(n) = \lceil \epsilon f(n) \rceil \text{ for all } n \in \mathbb{N}$$

Prove this theorem.

Solution: Let $L \in \mathrm{DSPACE}(f)$ and TM M that accepts L. As in the proof of Theorem §2.4 (linear speed-up) we compress the working tapes by factor $m = \epsilon^{-1}$ and simulate M on the compressed representation (and ensure optimal compression throughout by recompressing as necessary). Note that we do not compress the input onto a working tape (as this would

lead to problems for sublinear f). Since $L \in \mathsf{DSPACE}(f)$ the TM M has space consumption at most $f(n+2)$ for inputs of length n and our simulation has space consumption at most $\lceil \frac{f(n+2)}{m} \rceil$ (new cells).

$$\lceil \frac{f(n+2)}{m} \rceil = \lceil \epsilon f(n+2) \rceil = g(n+2)$$

Exercise 3.3

Prove the statement $\mathsf{DTIME}(O(f)) = \mathsf{DTIME}(f)$ for all $f\colon \mathbb{N} \to \mathbb{N}$ such that there exists $\epsilon > 0$ with $f(n) \geq (1+\epsilon)n$ for all $n \in \mathbb{N}$ (page 67 of 2nd lecture).

Solution: Let $L \in \mathsf{DTIME}(O(f))$. Then there exist TM M and $k, n_0 \in \mathbb{N}$ such that the time-consumption of M on inputs w with $n = |w| \geq n_0$ is at most $k \cdot f(n+2)$. Note that $n+2 \leq \frac{1}{(1+\epsilon)} f(n+2)$ by assumption (multiply both sides by $(1+\epsilon)$ to see this). Applying linear speed-up with factor $\frac{\epsilon}{(1+\epsilon)k}$ we obtain that the time-consumption is now at most

$$\lceil \frac{\epsilon}{1+\epsilon} f(n+2) \rceil + \underbrace{n+2}_{\leq \frac{1}{1+\epsilon} f(n+2)} \leq \lceil \frac{\epsilon}{1+\epsilon} f(n+2) \rceil + \frac{1}{1+\epsilon} f(n+2) \leq f(n+2) + 1$$

for all inputs of length $n \geq n_0$. We modify the TM such that it memorizes the correct acceptance behavior for all inputs of length shorter than n_0 (these are finitely many), so its time-consumption for such inputs w becomes $|w| + 2 \leq f(|w|+2)$. Hence $L \in \mathsf{DTIME}(f)$.

For completeness' sake, let us also take care of the final increment $(+1)$ in the chain of inequalities above. This is not very instructive, but demonstrates that it can be done. Applying linear speed-up with factor $\frac{\epsilon}{2(1+\epsilon)k}$ and requiring that $n_0 \geq 2\epsilon^{-1}$ we obtain that the time-consumption is now at most

$$\begin{aligned}
\lceil \frac{\epsilon}{2(1+\epsilon)} f(n+2) \rceil + \underbrace{n+2}_{\leq \frac{1}{1+\epsilon}\frac{2}{2} f(n+2)} &\leq \frac{\epsilon}{2(1+\epsilon)} f(n+2) + 1 + \frac{2}{2(1+\epsilon)} f(n+2) \\
&= \frac{2+\epsilon}{2(1+\epsilon)} f(n+2) + 1 \\
&= \frac{2+\epsilon}{2(1+\epsilon)} f(n+2) + \frac{\epsilon}{2(1+\epsilon)} (1+\epsilon) \underbrace{\frac{2}{\epsilon}}_{\leq n_0 \leq n+2} \\
&\leq \frac{2+\epsilon}{2(1+\epsilon)} f(n+2) + \frac{\epsilon}{2(1+\epsilon)} \underbrace{(1+\epsilon)(n+2)}_{\leq f(n+2)} \\
&\leq \frac{2+\epsilon}{2(1+\epsilon)} f(n+2) + \frac{\epsilon}{2(1+\epsilon)} f(n+2) \\
&= f(n+2)
\end{aligned}$$

for all inputs of length $n \geq n_0$.

Exercise 3.4

Recall the proof of the Theorem of Hennie and Stearns (pages 27–114 of 3rd lecture).
Why do we not simply call Simulate$(f(n))$?

Solution: In general we cannot compute $f(n)$ fast enough (since f is not necessarily time-constructible). To avoid computation of $f(n)$, we increase step-wise the number of simulated steps. Indeed f might not even be computable.

Exercise 3.5

Determine the time-consumption of Savitch's algorithm (page 135 of 3rd lecture).

Solution: We simplify the analysis by ignoring many simple operations. Let n be the input size, for which we assume $n \geq |V|$ and $n \geq |E|$ for input graph (V, E). Let t_i be the worst-case time-consumption of call Reach(v, i, v') for some $v, v' \in V$. For $i = 0$ let the time-consumption be $t_0 \in O(x)$ [linear time]. For $i \geq 1$ we have $t_i \in O(xt_{i-1})$; i.e. $t_i \in O(x^i \cdot x)$. In the initial call we have $i \leq \lceil \log n \rceil$ and thus obtain complexity $x^{O(\log x)}$, which proves super-polynomial growth. To see this perform change of basis to obtain basis 2

$$x^{O(\log x)} = (2^{\log x})^{O(\log x)} = 2^{O(\log x) \cdot \log x} = 2^{O(\log^2 x)}$$