Lehrstuhl für Informatik I
– Algorithmen und Komplexität –
Universität Würzburg

Würzburg, 18 January 2023

Johannes Zink
Oksana Firman

Exercise Sheet #11

Advanced Algorithms (WS 2022/23)

Exercise 1 – Warm-up

Construct a suffix tree and a suffix array for the string $\mathrm{mississippi}$. **6 Points**

Exercise 2 – Space requirement of suffix trees

Each edge of a suffix tree of a string T is labeled with an infix $T[i, j]$ of T. In the lecture, we stated that these labels should be implicitely encoded by storing the indices i, j to ensure that the tree requires only $\mathcal{O}(|T|)$ space. Construct an example where storing the labels explicitely (i.e., $T[i], T[i + 1], \ldots, T[j]$) requires superlinear space. **3 Points**

Exercise 3 – Counting queries

Let T be a string over an alphabet Σ. Describe a data structure to encode T such that all occurrences of a given pattern P can be *counted* in time $\mathcal{O}(|P| \log |\Sigma|)$. Your data structure should be constructable in time $\mathcal{O}(|T|)$. **3 Points**

Exercise 4 – Preprocessing multiple strings

Let $T_1, T_2, \ldots, T_\ell$ be strings of lengths $n_1, n_2, \ldots, n_\ell$ over a common alphabet Σ. Describe a data structure that encodes these strings such that all occurrences of a given pattern P (over alphabet Σ) in all of these strings can be reported in time independent of $n_1, n_2, \ldots, n_\ell$ and ℓ. Your data structure should be constructable in linear time.
4 Points

Exercise 5 – Longest common substring

Let T_1 and T_2 be strings over a common alphabet Σ. Design an algorithm to find the longest common substring (a string that occurs in both T_1 and T_2) in $\mathcal{O}(|T_1| + |T_2|)$ time.

Hint: Construct and traverse a suitably augmented suffix tree. **4 Points**

Please hand in your solutions on Wuecampus until the beginning of the next lecture, that is 14:15 on Wednesday, January 25.