CSE 549: Computational Biology	Instructor: Dominik Kempa

Homework 2

Stony Brook University, Spring 2022 — Posted: Feb 22, Due: March 6, 23:59 EST

Note: The submitted homework should be your own work. If a problem asks for a proof, you can use facts proved in lectures. Please submit your answers in one PDF, and all your code in a single separate file (no need to put source in the PDF), with compilation instructions in the comments. Please submit exactly two files (no archives).

Problem 1 [25]: Given a collection $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ of $k = \mathcal{O}(1)$ strings with total length $\sum_{i=1}^{k} |S_i| = n$, design an $\mathcal{O}(n)$-time algorithm to return the number of distinct strings that occur as substrings of exactly two elements of $\mathcal{S}$. For example, for $\mathcal{S} = \{\texttt{abaabb}, \texttt{abba}, \texttt{bbaaa}\}$, the algorithm should return 5 (the sought set of strings is $\{\texttt{ab}, \texttt{abb}, \texttt{bba}, \texttt{aa}, \texttt{baa}\}$). Write the pseudo-code of your solution (excluding construction of structures introduced in lectures), prove its correctness, and analyze its complexity.

Problem 2 [25]: Write the pseudo-code of the algorithm for the construction of suffix tree from the SA and LCP arrays in $\mathcal{O}(n)$ time. The resulting tree should be stored using the representation of tries presented in the lectures (i.e., using functions $\text{sdepth}(v)$, $\text{parent}(v)$, $\text{child}(v, c)$, and $\text{index}(v)$).

Problem 3 [25]: Let $T \in \Sigma^n$, where $\Sigma = [0\mathinner{.\,.}\sigma)$ for some $\sigma = \mathcal{O}(1)$. We define the *longest-previous-factor* array $\text{LPF}[1\mathinner{.\,.}n]$ such that $\text{LPF}[i] = \max\{\text{LCE}(i, i') : i' \in [1\mathinner{.\,.}i)\}$.

(a) [15] Design an $\mathcal{O}(n)$ time algorithm to compute the LPF array from T. Write the psedo-code, prove its correctness, and analyze its complexity. A solution with $\mathcal{O}(n \log n)$ running time will get about 10 pts.

(b)* [10] Prove that $\text{LPF}[1\mathinner{.\,.}n]$ is a permutation of $\text{LCP}[1\mathinner{.\,.}n]$.

Problem 4 [25]: Let T be a string of length n.

(a) [10] Show how to use the arrays computed by the KMR algorithm to compare any two substrings $T[i\mathinner{.\,.}i+\ell)$ and $T[j\mathinner{.\,.}j+\ell)$ of T (specified with the starting positions i and j, and the length ℓ) in $\mathcal{O}(1)$ time. The data structure should work for any ℓ (not just powers of two).

(b) [15] Use the above result to implement a deterministic data structure that can compare any two substrings in $\mathcal{O}(1)$ time. More precisely, write a program that reads a file with the same specification as in the previous homework. For each triple, output a word "YES" if it holds $T[i\mathinner{.\,.}i+\ell) = T[j\mathinner{.\,.}j+\ell)$, and "NO" otherwise. You can use the example file `in.txt` (from the previous homework) to test your program. Submit your program that computes the same output as in `out.txt`. *Note:* The files are provided for debugging rather than grading. The main goal of this problem is the implementation of KMR algorithm. Thus, the solution that passes the tests but uses a different algorithm (e.g., naive or Karp–Rabin) will not get any points.