CSE 549: Computational Biology Instructor: Dominik Kempa

Homework 1

Stony Brook University, Spring 2022 Posted: Jan 31, Due: Feb 17, 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 everything (including code) in one PDF.

Problem 1 [15]: For any positive integer p, we say that p is a *period* of a string S of length n, if for any $i \in [1\,..\,n-p]$ it holds $S[i] = S[i+p]$. Propose an algorithm that given a string S of length n, computes its smallest period in $\mathcal{O}(n)$ time. Write the complete pseudo-code of your solution. Prove its correctness and analyze its time complexity.

Problem 2 [15]: For any string S, let $\pi_S[1\,..\,|S|]$ denote the π array for string S, i.e., $\pi_S[i] = \max\{k \in [0\,..\,i) : S(0\,..\,k] = S(i-k\,..\,i]\}$. For each of the following claims, either prove that it is true, or provide a counterexample.

(a) Let S be a string of length m and assume that for some $i \in [1\,..\,m]$ it holds $\pi_S[i] = k$. Then, for any $j \in (i\,..\,m]$, it holds $\pi_S[j] \leq \pi_S[i] + (j-i)$.

(b) If $|S| = m$ and S can be written as $S = A^k$ for some string A (where A^k is defined as the concatenation of k copies of A, e.g., if $A = \texttt{aba}$ and $k = 2$, then $A^2 = \texttt{abaaba}$), then it holds $\pi_S[m] = m - |A|$.

(c) Let S be a string of length m and $c \in \Sigma$ be a symbol of the alphabet. By cS we denote the concatenation of c with the string S. For any $i \in [1\,..\,m]$ such that $\pi_{cS}[i+1] > 0$, it holds $\pi_S[i] = \pi_{cS}[i+1] - 1$.

Problem 3 [20]: Let T be a string of length n.

(a) Design a data structure of size $\mathcal{O}(n)$ that given any $i, j \in [0\,..\,n]$ such that $i \leq j$, computes the Karp-Rabin fingerprint of a substring $T(i\,..\,j]$, i.e., $\Phi(i,j)$ (see the formal definition of $\Phi(i,j)$ in the lecture slides) in $\mathcal{O}(1)$ time.

(b) Use the above result to implement a data structure that can compare any two substrings in $\mathcal{O}(1)$ time (by simply comparing two hashes). More precisely: write a program that reads a file containing the string (over English lowercase alphabet) in the first line, followed by a number k in the second line, followed by a sequence of k triples (i, j, ℓ) (in lines $3, \ldots, k+2$), and for each triple output a word (in a separate line) "YES" if it holds $T[i\,..\,i+\ell) = T[j\,..\,j+\ell)$, and "NO" otherwise. You can use the example file `in.txt` to test your program. In the example test file, it always holds $i, j \in [1\,..\,n]$ and $i+\ell, j+\ell \leq n+1$. Experiment with how large your need to make q so that there are no false positives. Submit your program with the value of q and r that computes the same output as in `out.txt`, i.e., the `diff` command of `out.txt` and the output of your program on `in.txt` should not show any difference. *Note:* The files are provided for debugging rather than grading. The main goal of this problem is the implementation of Karp-Rabin fingerprints. Thus, the solution that passes the tests but uses a different (e.g., naive) algorithm will not get any points.

Problem 4 [15]: Implement the KMP algorithm as a function that takes the text and the pattern as input, and returns the array with the starting positions of all occurrences of the pattern in the text. Then, write the code to test your function (by comparing the output of your KMP implementation to, e.g., the output of the naive algorithm) on millions of small

random instances (an input instance consists of a text and a pattern). Your program should test different string lengths and alphabet sizes. The result should be a single program. Your program will be checked by introducing some bug in your KMP function. Your testing program should detect that and show an instance on which the output of the bugged KMP implementation differs from the correct (naive) algorithm.

Problem 5 [15]: Let T be a string of length n. Suppose that the only way to learn anything about T is as follows: in $\mathcal{O}(1)$ time we can correctly answer a query $\mathrm{Eq}(i, j, \ell)$, that given any positions $i, j \in [1\,..\,n]$ and an integer $\ell \geq 0$ such that $i + \ell \leq n + 1$ and $j + \ell \leq n + 1$ returns true if and only if $T[i\,..\,i + \ell) = T[j\,..\,j + \ell)$. Design an algorithm to answer $\mathrm{LCE}(i, j)$ queries in $\mathcal{O}(\log n)$ time that, given $i, j \in [1\,..\,n]$, returns the length of the longest common prefix of $T[i\,..\,n]$ and $T[j\,..\,n]$. Write the pseudo-code, prove correctness, and analyze its complexity.

Problem 6 [20]: Let T be a string of length n and let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be a set of patterns. Let $\|\mathcal{P}\| := \sum_{i=1}^{k} |P_i|$ be the total length of patterns in $\mathcal{P}$. Show how to generalize the KMP algorithm from a single pattern to k patterns, so that:

- Preprocessing of $\mathcal{P}$ takes $\mathcal{O}(\|\mathcal{P}\|)$ time.
- All occurrences of patterns in $\mathcal{P}$ in the text T can be listed in $\mathcal{O}(n + \|\mathcal{P}\| + \mathrm{occ})$ time (where occ denotes the total number of reported occurrences) and similarly as in KMP, the text is streamed from left to right (i.e., you cannot preprocess the text).
- For simplicity, you can assume that all strings are over alphabet of size $|\Sigma| = \mathcal{O}(1)$.

Write the pseudo-code of your solution, prove its correctness, and analyze its complexity. *Hint*: Store $\mathcal{P}$ using trie, and appropriately generalize π.