CSE 549: Computational Biology	Instructor: Dominik Kempa
Homework 3	
Stony Brook University, Spring 2022	Posted: March 30, Due: April 10, 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: (1) PDF, (2) a single source file (no archives).

Problem 1 [25]: Let $\mathrm{ED}(X,Y)$ denote the standard edit distance between X and Y (allowing insertions, deletions, and substitutions). Let $\mathrm{LCS}(X,Y)$ denote the length of the longest common subsequence of X and Y. Let XY denote the concatenation of strings X and Y. For each of the following claims, either prove that it is true, or provide a counterexample.

(a) [8] For every $X, Y \in \Sigma^*$, it holds $\mathrm{LCS}(XX, YY) = 2 \cdot \mathrm{LCS}(X,Y)$.
(b) [8] For every $X, Y \in \Sigma^n$, it holds $2\mathrm{LCS}(X,Y) + \mathrm{ED}(X,Y) = 2|X|$.
(c) [9] For every $X, Y, Z \in \Sigma^*$, it holds $\mathrm{ED}(X,Z) \le \mathrm{ED}(X,Y) + \mathrm{ED}(Y,Z)$.

Problem 2 [25]: Design an $\mathcal{O}(n^2)$-time and $\mathcal{O}(n)$-space algorithm that, given a string $S \in \Sigma^n$, computes the minimal number of symbol insertions into S so that it is a palindrome, e.g., for $S =$ `cabae`, the answer is 2 since by inserting 2 characters into S we can modify S into `ecabace` or `ceabaec`, and it is not possible to turn S into a palindrome by inserting fewer than two symbols. Write the pseudo-code of your algorithm, prove its correctness, and analyze its complexity.

Problem 3 [25]: Let $X, Y \in \Sigma^n$. Denote $P(X,Y) = \{(i,j) : X[i] = Y[j]\}$, e.g., for $X =$ `aba` and $Y =$ `bac`, we have $P(X,Y) = \{(1,2),(2,1),(3,2)\}$. Assume that given $X, Y \in \Sigma^n$, we can compute the $\mathrm{LCS}(X,Y)$ (the length of the longest common subsequence of X and Y) in $\mathcal{O}((n+r)\log n)$ time, where $r = |P(X,Y)|$. Given any array $Q[1\,..\,n]$ of n nonnegative integers (Q may contain duplicates), describe an algorithm that returns the length (denoted as $\mathrm{LIS}(Q)$) of the longest increasing subsequence of Q in $\mathcal{O}(n \log n)$ time. For example, for $Q = [4,1,4,2,7]$ the algorithm should return $\mathrm{LIS}(Q) = 3$ (corresponding to the subsequence $1,2,7$ or $1,4,7$). You can either write the pseudo-code or describe the algorithm in words. Prove the correctness of your algorithm and analyze its complexity.

Problem 4 [25]:

(a) [10] Let $X \in \Sigma^m$ and $Y \in \Sigma^n$. Assume $n \le m$. Show how to compute the value $\mathrm{ED}(X,Y)$ (without the alignment) in $\mathcal{O}(mn)$ time and using only $n + \mathcal{O}(1)$ words of extra space (on top of the read-only arrays containing X and Y). Write the pseudo-code of your algorithm, analyze its complexity, and prove its correctness.
(b) [15] Implement the above algorithm, i.e., write a program that, given the file with the same specification as in the previous two homeworks, for each triple (i, j, ℓ) outputs the value $\mathrm{ED}(T[i\,..\,i+\ell), T[j\,..\,j+\ell))$ in a separate line. You can use the example file `in.txt` (from previous homeworks) to test your program. Submit your program that computes the same output as in `out.txt` (provided along with this homework). A correct implementation using $n + \mathcal{O}(1)$ space will get 100% of the points. A solution that uses $\mathcal{O}(n)$ (but not $n + \mathcal{O}(1)$) space will get 50% of the points. Solutions that use more space will

not get any points. Solution for part (b) of this problem is graded separately from part (a), i.e., the code for part (b) does not automatically solve part (a).

Please pay attention to the format in which you need to submit your code (see the instructions at the beginning of this PDF). Solutions submitted in the wrong format automatically lose 25% of points.