This homework is due Wednesday, April 18, at the start of class. The questions are on topics in sequence comparison.

The homework is worth a total of 100 points. In problems with several parts where point breakdowns are not given, each part has equal weight.

When grading the homework, only a subset of two problems will be graded, whose points together add up to a total of 100 points.

Please write only on one side of the paper, start each problem on a new page, and staple the problems in order. Conciseness counts!

(1) **(Edit distance between circular strings) (50 points)** Biological sequences are sometimes circular, such as the genomes of bacteria or organelles like mitochondria. In general, a circular string $A[1 : m]$ has its letters $A[1], A[2], \ldots, A[m]$ arranged in order around a circle (with no start or end), so that $A[m]$ is immediately followed by $A[1]$. This problem asks you to generalize the computation of edit distance to circular strings.

Given two circular strings $A$ and $B$ of lengths $m$ and $n$, design an algorithm to compute the edit distance between $A$ and $B$, under a well-behaved edit cost function $\delta$, in $O(mn \log \min\{m, n\})$ time.

(Note: This problem shows edit distance for circular strings can be computed with just a logarithmic-factor increase in running time. Conceptually, finding the edit distance between circular strings $A$ and $B$ is equivalent to minimizing ordinary edit distance over all possible ways of linearizing $A$ and $B$ into non-circular strings. More formally, a cyclic shift of string $A[1 : m]$ is the non-circular string obtained by the concatenation $A[i : m] \cdot A[1 : i - 1]$ for some $1 \leq i \leq m$. The edit distance between circular strings $A$ and $B$ is then the minimum edit distance between all cyclic shifts of $A$ and all cyclic shifts of $B$.)

(Hint: It suffices to treat $A$ and $B$ as ordinary, non-circular strings, and to compare this fixed string $A$ against all $n$ cyclic shifts of $B$. To handle all these shifts of $B$ simultaneously, consider the alignment grid graph $G$ between $A$ and the concatenation $B \cdot B$. Within this graph $G$, the edit distance between $A$ and a given cyclic shift of $B$ corresponds to a shortest path between a particular source vertex $s$ and a particular sink vertex $t$. Each of the $n$ cyclic shifts corresponds to a shortest path between a different source and sink. Show how to solve these $n$ different source-sink shortest path problems in $G$ without solving each one from scratch.)

(2) **(Tandem repeats of a fixed pattern) (50 points)** Biological sequences such as genomes often have embedded within them tandem repeats, which are multiple consecutive copies of a short string $w$, where the copies in the repeat can have errors, such as the substitution, insertion, and deletion of characters. Formally, a tandem repeat of string $w$ is $i$ consecutive copies of $w$ for some $i \geq 1$, written $w^i$. This problem asks you to find the best local alignment between a given long string, and tandem repeats of a given short string, where the number of copies in the occurrence of the tandem repeat within the longer string is not known in advance.

Let pattern $P$ and text $T$ be strings of lengths $m$ and $n$, and let similarity function $\sigma$ assign scores to substitutions, insertions, and deletions. Design an algorithm that, over all substrings $\tilde{T}$ of $T$, and all substrings $\tilde{P}$ of all tandem repeats $P^i$ for $i \geq 1$, finds the maximum-similarity local alignment between $\tilde{T}$ and $\tilde{P}$ under similarity function $\sigma$, in $O(mn)$ time.
(Hint: Show how to solve this problem by finding a longest path through a grid graph with $O(mn)$ vertices that contains cycles, and find a way to compute the longest path in $O(mn)$ time.)

(3) **(Simplified RNA folding)** (50 points) Let $\Sigma = \{a, c, g, u\}$ be the alphabet of RNA sequences, and $S$ be an RNA sequence. For every letter $c \in \Sigma$, there is a *complementary* letter $\overline{c} \in \Sigma$, given by $\overline{a} = u$, $\overline{u} = a$, $\overline{c} = g$, and $\overline{g} = c$. When an RNA molecule described by sequence $S$ folds into a three-dimensional structure, chemical bonds form between pairs $(i, j)$ of distinct positions in $S$, where we may assume $i < j$, subject to the physical constraints that paired letters are complementary,

$$S[i] = \overline{S[j]},$$  

and that for all paired positions $(i, j)$ and $(\overline{i}, \overline{j})$, either the intervals for the pairs are disjoint,

$$i < j < \overline{i} < \overline{j},$$  

or properly nest,

$$i < \overline{i} < \overline{j} < j.$$  

A set of pairs $\mathcal{P}$ is *valid* if it satisfies conditions (1) through (3). Note that in a valid pairing, a position will be involved in at most one pair (and might not be paired with anyone).

Given a weight function $\omega : \{(c, \overline{c}) : c \in \Sigma\} \rightarrow \mathbb{R}^+$ that weights pairs of complementary letters, and an RNA sequence $S$ of length $n$, design an algorithm that, by dynamic programming, finds a valid pairing $\mathcal{P}$ for $S$ of maximum total weight $\sum_{(i,j) \in \mathcal{P}} \omega(S[i], S[j])$, in $O(n^3)$ time.

(Note: Be sure to follow the four-step framework for designing dynamic programming algorithms, namely: (1) characterize the recursive structure of an optimal solution, (2) derive a recurrence equation for the value of an optimal solution to a subproblem, (3) evaluate the solution value for the optimal solution to the input problem bottom-up in a table, and (4) recover the optimal solution from the table of solution values.)

(Hint: Derive a recurrence for the maximum total weight of a pairing for every substring $S[i:j]$ of $S$.)