This homework is due Wednesday, February 28, at the start of class. The questions are on designing string algorithms for new problems, using suffix arrays, their associated height array, and the Ferragina-Manzini index.

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) **(Minimum cover)** (50 points) Given strings $A$ and $B$, a minimum cover of $A$ by $B$ is a decomposition $A = w_1 w_2 \cdots w_k$ where each $w_i$ is a substring of $B$ and $k$ is minimum. In other words, a minimum cover of $A$ by $B$ expresses string $A$ as the concatenation of the fewest possible substrings of $B$, where these substrings can come from anywhere in string $B$.

Design an algorithm that computes a minimum cover (if one exists) of string $A$ by string $B$, where these strings have lengths $m$ and $n$, in $O((m+n) \log(m+n))$ time. Be sure to argue why your algorithm is correct.

(Hint: Construct a suffix array and its height array for an appropriate string, and use a greedy strategy for finding the cover.)

(Note: It is actually possible to find a minimum cover in $O(m+n)$ time, which is optimal.)

(2) **(Longest suffix-prefix overlaps)** (50 points) Suppose we have a collection of $k$ strings $S = \{S^{(1)}, S^{(2)}, \ldots, S^{(k)}\}$. For an ordered pair of strings $(A, B)$, their suffix-prefix overlap is the longest exact match between a suffix of $A$ and a prefix of $B$. For each string $A$ in $S$, we would like to know both: (i) the longest overlap $(A, B)$ that $A$ has over all other strings $B$ in $S$, and similarly (ii) the longest overlap $(B, A)$ that $A$ has over all other strings $B$. Note that for $k$ strings, this is $2k$ pieces of information: for each string $A$, the longest overlap with a suffix of $A$, and the longest overlap with a prefix of $A$.

(a) Given a collection $S$ of $k$ strings of total length $n$, design an algorithm that determines these longest suffix-prefix overlaps for all $k$ strings in $O(n \log n)$ time, using suffix arrays and height arrays. Be sure to argue why your algorithm is correct.

(Hint: Construct one suffix array for all the strings in $S$, and make use of its height array. You may find it useful to do the same for the reverse of all the strings.)

(b) Now design an algorithm that solves this same problem of finding these longest suffix-prefix overlaps for all strings in $O(n)$ time, using the Ferragina-Manzini index instead of a suffix array.

(Note: This yields an optimal algorithm for longest suffix-prefix overlaps.)

(3) **(Longest nonoverlapping repeat)** (50 points) A nonoverlapping repeat in a string $S$ is a string $w$ such that $S = xwywz$, where $x$, $y$, and $z$ are possibly empty strings. (In other words, in a nonoverlapping repeat, the two occurrences of the repeat do not overlap.)

Design an algorithm that finds a longest nonoverlapping repeat in a string $S$ in $O(n)$ time, where $n$ is the length of $S$. Be sure to argue why your algorithm is correct.

(Hint: Use a suffix array and its corresponding height array.)
(4) (bonus) **(Longest common substring of three strings)** (10 points) For a collection of strings, their *longest common substring* is another string that is a substring of all the strings in the collection, and has greatest length.

Given three strings $X$, $Y$, and $Z$, design an algorithm that finds the longest common substring of $X$, $Y$, and $Z$ in $\Theta(n)$ time, where $n = |X| + |Y| + |Z|$.

(Hint: Construct a suffix array and height array for a single string, and generalize the approach for finding the longest common substring of two input strings.)

(5) (bonus) **(Maximal repeats)** (20 points) A *maximal repeat* in a string $S$ is a triple $(i, j, \ell)$ such that $S$ contains a repeat of length $\ell$ starting a positions $i$ and $j$, and this repeat cannot be extended further to the left or right. Formally, $S[i : i + \ell - 1] = S[j : j + \ell - 1]$, but $S[i - 1] \neq S[j - 1]$ and $S[i + \ell] \neq S[j + \ell]$.

Given a string $S$ of length $n$ and an integer $k$, design an algorithm that finds the $k$ longest maximal repeats in $S$ in time $O(n + k)$, using a suffix array.

Note that Problems (4) and (5) are a *bonus* questions. They are not required, and their points are not added to regular points.