This homework is due Thursday, October 5, at the start of class. The questions are on designing string algorithms for new problems, using suffix arrays and their associated height array.

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

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) (Longest common prefix lengths from heights) (20 points) This question asks you to prove a key relationship on height array values that is used in many string algorithms.

The longest common prefix of two strings $X$ and $Y$ is a string that is a prefix of both $X$ and $Y$ and has greatest length. Denote the length of the longest common prefix of $X$ and $Y$ by $lcp(X,Y)$.


Prove that for any two suffixes of $S$, starting at positions $A[i]$ and $A[j]$ with $i < j$, $lcp(S_{A[i]}, S_{A[j]}) = \min_{i < k \leq j} \{H[k]\}$.

(Hint: Show that the right-hand side of the above equation is both a lower- and an upper-bound on the left-hand side.)

(2) (Minimum cover) (40 points) Given strings $A$ and $B$, a minimum cover of $A$ by $B$ is a decomposition $A = w_1w_2\cdots w_k$ where each $w_i$ is a substring of $B$ and $k$ is minimum.

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. 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: This problem can actually be solved in $O(m+n)$ time using a suffix array and its height array.)

(3) (Longest suffix-prefix overlaps) (40 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 $S^{(i)}$ in $S$, we would like to know the longest suffix-prefix overlap $(S^{(i)}, S^{(j)})$ that $S^{(i)}$ has over all other strings $S^{(j)}$, and similarly the longest overlap $(S^{(j)}, S^{(i)})$ that $S^{(i)}$ has over all other strings $S^{(j)}$. Note that for $k$ strings, this is $2k$ pieces of information: two longest overlaps for each string $S^{(i)}$.

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. 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.)

(Note: This problem can actually be solved in $O(n)$ time using the Ferragina-Manzini Index instead of a suffix 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 described in class for finding the longest common substring of two input strings.)

Note that Problem (4) is a bonus question. It is not required, and its points are not added to regular points.