This homework is due Wednesday, February 14, at the start of class. The questions concern topics in constructing and using suffix arrays.

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) (Preserving lexicographic order on suffixes) (50 points) Let $S[1:3n]$ be a string whose length is a multiple of 3 that is over an alphabet $\{1, 2, \ldots, k\}$, and let $\tilde{S}[1:2n]$ be the shorter string corresponding to the recursive subproblem solved by the algorithm of Kärkkäinen and Sanders for constructing a suffix array. For a string $X$ of length $m$, denote the suffix of $X$ starting at position $i$ by $X_i := X[i:m]$. For a position $i$ in $S$ where $i \not\equiv 0 \pmod{3}$, let $I(i)$ be the corresponding position in $\tilde{S}$.

Prove that for all positions $i, j$ in $S$ where $i, j \not\equiv 0 \pmod{3}$, 

$S_i \preceq S_j$ if and only if $\tilde{S}_{I(i)} \preceq \tilde{S}_{I(j)}$.

(Hint: As part of the proof, show that because $\tilde{S}$ is constructed from $S$ with two 0’s appended, lexicographic comparisons of suffixes of $\tilde{S}$ will not pass the midpoint of $\tilde{S}$ that separates positions $I(i)$ with $i \equiv 1 \pmod{3}$ from positions $I(j)$ with $j \equiv 2 \pmod{3}$.)

(2) (Longest-common-prefix lengths from heights) (50 points) 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 $\text{lcp}(X,Y)$.


Prove that for any two suffixes of $S$, starting at positions $A[i]$ and $A[j]$ with $i < j$, 

$$\text{lcp}(S_{A[i]}, S_{A[j]}) = \min_{i \leq k < j} \{H[k]\}.$$ 

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

(3) (Interval-minimum queries) (50 points) Suppose you are given an array $A[1:n]$ of real numbers. An interval-minimum query on $A$ is: given an interval $[i,j]$ where $1 \leq i \leq j \leq n$, compute 

$$\min_{i \leq k \leq j} \{A[k]\}.$$ 

Design an algorithm that, after spending $\Theta(n)$ time preprocessing $A$, can answer any interval-minimum query on $A$ in $O(\log n)$ time.

Be sure to argue that your algorithm is correct, and analyze the running time of your algorithm.

(Hint: Make use of a balanced binary tree, augmented with additional information.)
(4) **(Faster interval-minimum queries) (bonus)** (10 points) Interval-minimum queries, defined in Problem (3) above, can be solved even faster than $O(\log n)$ time per query.

Design an algorithm that now uses $\Theta(n \log n)$ time and space preprocessing, but afterward can answer any interval-minimum query in $O(1)$ time.

(Hint: At each position $i$ in the input array $A[1:n]$, consider precomputing the solution for query intervals starting at $i$ of lengths $1, 2, 4, 8, 16, \ldots$, in powers of two. Then figure out how to use these precomputed solutions to answer an arbitrary query in constant time.)

(Note: This problem can actually be solved in $O(1)$ time per query, using $\Theta(n)$ time and space preprocessing—which is optimal—by a very sophisticated algorithm.)

(5) **(Single-pair longest common substring)** (50 points) Given two strings $S$ and $T$, a **common substring** of $S$ and $T$ is a string $w$ that is a substring of both $S$ and $T$.

Design an algorithm that finds the longest common substring of two strings of lengths $m$ and $n$ in $\Theta(m + n)$ time. Be sure to argue that your algorithm is correct.

(Hint: Construct a suffix array for an appropriate string, and use its height array.)

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