This homework is due Thursday, September 21, 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.

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) (Properties of lexicographic order) (20 points) The correctness of the algorithm for string matching that uses a suffix array relies on two properties of the length-limited lexicographic order relationship $\preceq_k$ on strings. This question asks you to prove that these properties hold.

(a) (10 points) Prove that $\preceq_k$ is transitive: namely, that for any three strings $A$, $B$, and $C$,

$$A \preceq_k B \text{ and } B \preceq_k C \implies A \preceq_k C.$$  

(Hint: First try proving transitivity for ordinary lexicographic order $\preceq$, which is not length-limited by $k$.)

(b) (10 points) Prove that $\preceq_k$ is anti-symmetric: specifically, that for any two strings $A[1:m]$ and $B[1:n]$,

$$A \preceq_k B \text{ and } B \preceq_k A \text{ if and only if } A[1:\tilde{k}] = B[1:\tilde{k}],$$

where $\tilde{k} := \min \{k, \max \{m, n\}\}$.

(Hint: Again first try proving anti-symmetry for ordinary lexicographic order $\preceq$, which is not length-limited by $k$, or equivalently where $k = \max \{m, n\}$.)

(2) (Preserving lexicographic order on suffixes) (20 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 $i, j \not\equiv 0 \pmod{3}$,

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

(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 $j \equiv 2 \pmod{3}$.)

(3) (Interval-minimum queries) (30 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 search tree, augmented with additional information.)

(Note: This problem can actually be solved in $O(1)$ time per query.)

(4) **(Leftmost occurrence)** (30 points) When solving string matching, sometimes we might not want to spend the time required to output all occurrences of the pattern in the text, but instead only output a single occurrence, along with a count of the total number of occurrences. Given a text string $S$ and a pattern string $P$, the *leftmost occurrence* of $P$ in $S$ is an occurrence whose start position of the substring of $S$ that matches $P$ is smallest.

Design an algorithm that both:

- reports the *leftmost occurrence* of $P$ in $S$, and
- outputs the *number of occurrences* of $P$ in $S,

in $O(m \log n)$ time per query, after spending $O(n)$ time preprocessing $S$, where $m$ is the length of $P$ and $n$ is the length of $S$.

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

(Hint: Make use of interval-minimum queries.)