(1) 1. Given s and t, we want to compute the length of their longest common **substring**. Show how to solve this problem in linear time given a linear time algorithm for construction the suffix array and the lcp array. What about the longest string that appears at least twice in s but does not appear in t?

(1) 2. Show how to compute the lcp array in $\mathcal{O}(n)$ time using the following property (that you should prove): for any suffix $w[i..n]$, $\text{lcp}[SA^{-1}[i]] - 1 \leq \text{lcp}[SA^{-1}[i+1]]$, assuming $i < n$ and neither $SA^{-1}[i]$ nor $SA^{-1}[i+1]$ is the first element of the suffix array.

(0.5 or 1) 3. Given a permutation π, how to check in $\mathcal{O}(n)$ time if there exists a string w such that its suffix array SA_w is the given permutation? Partial credit for a solution which only checks if there is such a binary string.

(2) 4. Our RMQ structure needs access to the original array. Design a (more complicated, but following the same general idea) structure that uses only $2n + o(n)$ bits, and answers queries in constant time **without** accessing the original array. Partial credit for using $\mathcal{O}(n)$ bits.

(0.5) 5. Show that $2n + o(n)$ is necessary in the above question.

(1) 6. Show how to implement select queries in constant time with only $\mathcal{O}(n \log\log n / \log n)$ additional bits of space. Hint: it's more of the same.