1. Show how to compute the lcp array, given a word w and its corresponding SA array, in linear time. Recall that $\mathrm{lcp}[i]$ is the length of the longest common prefix of $w[\mathrm{SA}[i-1]..n]$ and $w[\mathrm{SA}[i]..n]$, for $i = 2, 3, \ldots, n$.

2. Given two strings s and t, we are interested in computing the length of their longest common substring, which is a string occurring in both s and t. Show how to solve this problem in linear time using the suffix array. You can assume that the lcp array is available, too.

3. Given a permutation on $\{1, 2, \ldots, n\}$, we want to find a word $w \in \Sigma^n$ such that its suffix array SA_w is exactly the given permutation.
   (a) Show a linear time algorithm solving the problem for $\Sigma = \{a, b\}$.
   (b) For extra credit: show a linear time algorithm solving the problem for $\Sigma = \{a, b, \ldots, z\}$.

(2 points) 4. A set $B \subseteq [0, t)$ is called a difference cover modulo t when, for every $x, y \in [0, t)$, there exists $\delta \in [0, t)$ such that $x + \delta, y + \delta \in B$. Construct a difference cover of size $\mathcal{O}(\sqrt{t})$. What is the smallest c such that you can obtain a difference cover of size $c\sqrt{t} + \mathcal{O}(1)$?

5. Assume that you are given a word w and its suffix array (with the lcp information). Show how to count the number of **different** nonempty subwords of w, i.e., calculate $|\{s[i..j] : 1 \leq i \leq j \leq |w|\}|$ in $\mathcal{O}(n)$ time. For instance, $aaab$ has 7 different subwords.