- This problemset has *two* questions.
- To get the credit for questions marked as SPOJ, you must get them accepted on `http://www.spoj.com/AOS`, but you **don't** have to send any explanation!
- For other questions, either send the solutions to `gawry1+aos@gmail.com`, or leave them in the envelope attached to the doors of my office (room 321).

1. 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.

2. The goal of this (long) problem is to show that in the RMQ problem we can actually reduce the general case to the special one where $|A[i+1] - A[i]| \leq 1$, which we call *bounded* RMQ. Recall that RMQ is to preprocess an array $A[1..n]$ so that we can find the **position** of the minimum in any $A[i..j]$ efficiently.
   (a) Lowest common ancestor problem (or LCA) is to preprocess a rooted tree on n nodes so that given any two nodes we can find their lowest common ancestor (lowest means closest to the nodes). First show how to preprocess the tree in linear space so that given two nodes you can decide whether one is an ancestor of the other in constant time (think about preorder numbers using in the depth-first search).
   (b) Design an $\mathcal{O}(n \log n)$ time and space preprocessing algorithm which allows computing the LCA of any two nodes in $\mathcal{O}(\log n)$ time using a binary search-like procedure (use the doubling and powers-of-two trick).
   (c) Consider the following procedure: start at the root and traverse the whole tree in a depth-first fashion, at each step moving either one edge down or one edge up. Prove that the total number of steps is $\mathcal{O}(n)$. Then consider the sequence of numbers $1, -1$ denoting whether you went one edge down or up. What is the relation between the LCA of two nodes and an array of prefix sums of your sequence? Observe that this relation allows you to reduce LCA on a tree of size n to bounded RMQ on an array of length $\mathcal{O}(n)$.
   (d) The *Cartesian tree* of a given array $A[1..n]$ is recursively defined as follows: choose the minimum element $A[i]$ and make it the root. Then recurse on $A[1..i-1]$ and make the resulting tree the left child of the root, and recurse on $A[i+1..n]$ and make the resulting tree the right child of the root. Show how to construct the Cartesian tree in $\mathcal{O}(n)$ time by starting with the empty tree, and then constructing the trees for $A[1..1]$, $A[1..2]$, $A[1..3]$, ..., $A[1..n]$ (this is probably the most complicated part).
   (e) Find a relation between RMQ query about $A[i..j]$ and a LCA query on the corresponding Cartesian tree. Observe that this relation allows you to reduce RMQ on an array of length n to LCA on a tree of size n.
   (f) Finally, combine the observations to show that RMQ on an array of length n can be reduced to bounded RMQ on an array of length $\mathcal{O}(n)$.