2nd tutorial

A refresher of definitions:

- The KMP (Knuth-Morris-Pratt) automaton is at state α after processing a string σ iff α is the longest suffix of σ which is also a prefix of the needle ι.
- When the KMP automaton is at state α and does not match the next character, it takes a back edge $b(\alpha)$ which goes to the longest proper suffix of α which is also a prefix of ι.

1. KMP. Construct the search automata for the words "kokos" and "ananas". We will see in the lecture how to do this in linear time, but for now just use the definition of the failure function, which is:
For a state α, the back edge $b(\alpha)$ goes to the longest proper suffix of α which is a prefix of ι.

2. Rotation. A rotation of a string α is the string $\alpha[K:]\alpha[:K]$ for some $K, 0 \leq K < |\alpha|$, e.g. "nasana" is a rotation of "ananas" for $K = 3$.
Design an algorithm which decides whether α' is a rotation of α.

3. Ordered trees. Say that a tree is *ordered* if it is rooted and for each node, there is an order on its children (i.e., we can speak of the first, second, third etc. child). A subtree of an ordered tree is obtained by choosing a node, taking its subtree, and then possibly also removing some edges coming from the root, but only from the left and the right, that is, the edges which remain have to form a contiguous sequence of children (e.g., if the chosen node as 5 children, we can remove the first, fourth, and fifth, but not the first, third, and fifth).
Given two ordered trees, decide whether one is a subtree of another.

4. Periodicity. How to decide whether a word α is *periodic*? By this we mean that there exists a word β and a number $k > 1$ s.t. $\alpha = \beta^k$, that is, the concatenation of k copies of β.

5. Palindrome. Denote by α^{-1} the reversed string α, that is, $\alpha^{-1}[i] = \alpha[-i]$ (in Pythonic notation). A string α is a *palindrome* if $\alpha = \alpha^{-1}$. Design an efficient algorithm that computes the length k of the longest palindromic prefix of α, i.e. $\alpha[:k] = (\alpha[:k])^{-1}$.

6. Too many occurences. Find an example input of patterns and text which has asymptotically more than linear number of occurences. Specifically, for every n, show that there is input to the TEXT SEARCH problem such that $|T| + \sum_i |P_i|$ is $\Theta(n)$, and the number of occurences is not $\mathcal{O}(n)$.

7. Counting words. You are given a string S of length k and a number n. How to efficiently compute the number of strings of length n over the alphabet $\{\mathtt{a}, \ldots, \mathtt{z}\}$ which do not contain S as a substring?