Problem 1. The construction of suffix trees given in class assumes that the alphabet is of constant size, that is $|\Sigma| = O(1)$. Explain how Ukkonen’s algorithm can be modified for the case where the alphabet size is not constant. More specifically, show how to construct a suffix tree for a string of length $m$ in time $O(m \log |\Sigma|)$, and how to perform a query for a string of length $n$ in time $O(n \log |\Sigma|)$.

Problem 2. Let $\Sigma$ be an alphabet of constant size. Recall that the Substring Problem for a Database is as follows. The input consists of a set $T$ of strings with alphabet $\Sigma$ and of total length $m$. The goal is to preprocess $T$ so that given a query string $P \in \Sigma^n$, we can find all occurrences of $P$ in all strings in $T$.

The solution we discussed in class for this problem was the following. Let $T = \{T_1, \ldots, T_k\}$. We build a suffix tree for the string $S = T_1$s_1T_2$s_2\ldots T_k$s_k$, where the symbols $s_1, \ldots, s_k$ denote distinct characters that are not in $\Sigma$. Using the solution of Problem 1 this idea can be implemented with alphabet $\Sigma' = \Sigma \cup \{s_1, \ldots, s_k\}$, which is of size $k + O(1)$. This results in preprocessing time $O(m \log k)$ and query time $O(n \log k)$.

Show how to solve this problem with preprocessing time $O(m)$ and query time $O(n)$.

Problem 3. Let $\Sigma$ be an alphabet of constant size. A substring $P \in \Sigma^*$ is called a prefix repeat of a string $S \in \Sigma^*$ if $P$ is a prefix of $S$ and it is of the form $P = QQ$ for some string $Q \in \Sigma^*$. Give a linear-time algorithm to find the longest prefix repeat of an input string $S$. 