Self-Practice Week 9 - Strings (part 1)

String Algorithms

The goal of this assignment is to develop a better understanding of the trie data structure and basic algorithms for efficient string sort & search.

Exercise 1 – Longest common prefix

Given a set of N strings, find the longest common prefix. For example, given in input set S: {"apple", "appointment", "appendix", "appeal", "apparel"}, your algorithm should return "app".

Hint: insert all N strings in a trie, then perform a walk-through.

Exercise 2 – Pattern search

Given a text T of length N, and a pattern P of length M < N, implement the Knuth-Morris-Pratt algorithm to find all occurrences of P in T. Experimentally compare the performance of this algorithm against a simple brute force pattern search algorithm, as you vary your input (for an example of real large text, use file "5-mobydick.txt"; for worst-case scenario, generate your own input).

Exercise 3 – Longest common substring

Given two strings, find the longest common substring (not necessarily a prefix). For example, given in input strings "appendix" and "appeal", their longest common substring is "appe" (which is coincidentally also a prefix), whereas given in input strings "appendix" and "compendium", their longest common substring is "pendi".

Hint: a naïve solution would yield $O(M^2)$ complexity (with M being the length of the longest substring). Read about suffix trees (chapter 6) to learn a new data structure that would support this kind of operation in linear time.