Mandatory exercise: Compressed Pattern Matching

Patrick Hagge Cording

1 Compressed subsequence recognition revisited. A string S' is a subsequence of S if S' can be obtained by deleting characters from S.

In one of the exercises you designed an algorithm whose running time is $O(n\sigma + M \log N \log\log N)$, i.e., it was depending on the alphabet size σ. In some cases, σ is very large so now we want to design an algorithm with running time and space usage independent of σ.

You are given an SLP of size n compressing a string S of size N and a pattern P of size M. Give an algorithm that determines if P is a subsequence of S. Your algorithm should run in $O(nM)$ time and use $O(nM)$ space. Similarly to the simple pattern matching algorithm shown in the lecture, your algorithm should not use any auxiliary primitives or data structures as for instance random access or any tree data structures.