PAL labs 9

16 / 11 / 2022

Let's denote Levenshtein distance of two words x and y as $d(x, y)$. We know that for three words u, v, w it holds that $d(u, v) = d_1$, $d(v, w) = d_2$. Decide upper and lower bound on the value of $d(u, w)$ w.r.t. d_1, d_2? There is only one alphabet for all the words.

Let's denote Hamming distance of words v and w of the alphabet A as $HD(v, w)$, similarly, denote Levenshtein distances as $LD(v, w)$. Decide which of the following cases may happen and if a case may happen, show an example of two words v and w of length at least 5 such that an (in)equality holds.

a) $HD(v, w) < LD(v, w)$,

b) $HD(v, w) = LD(v, w)$,

c) $HD(v, w) > LD(v, w)$.

There are two finite sets of strings, M_1 and M_2, over the alphabet A. Describe how you wold construct a finite automaton accepting all words w over the alphabet A such that at least one prefix of the word w is a member of M_1 and at least one suffix of w is a member of M_2. Recall that a whole word is considered to be its own prefix and suffix. Construct an example for $|M_1| = |M_2| = 2$.

Find in the text T all occurences of the substrings which Hamming distance form the pattern P is at most k. Apply the dynamic programming approach. a) T = *ccacbaabccacc*, P = *abcba*, k = 2

Find in the text T all occurences of the substrings which Levenshtein distance form the pattern P is at most k. Apply the dynamic programming approach.

b) T = 0100111010000010101000 , P = 11100, k = 1

Construct an NFA which finds each word from the set M in a text over the alphabet A.

a) $A = \{a, b, c\}$, $M = \{a, b, ba, bc, aaa, bab, ccc, abbc, abcc\}$.

Construct a DFA which finds each word from the set M (from the previous task) in a text over the alphabet A.

We say that two string X and Y over an alphabet have reduced Levenshtein distance equal to k if and only if k is the minimal number of edit operation to get X from Y. Insert and Delete are the only allowed edit operations. Describe an algorithm which will compute reduced Levenshtein distance of X and Y using dynamic programming.