Dynamic Programming Prac

Last modified on Sunday, 14 October 2018 by f.maire@qut.edu.au

The ***Levenshtein distance*** is a string metric for measuring the difference between two sequences. Its variations can be found in many domains including intelligent editors, bioinformatics and robotics (for place recognition).

Informally, the Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other. It is named after Vladimir Levenshtein, who considered this distance in 1965.

The recurrence relations for the Levenshtein distance can be found in the lecture slides on Dynamic Programming (see slides 29 to 33). In this practical, we will assume that the costs of the different edit operations (insertions, deletions and substitutions) are not identical.

Exercise 1

- The Levenshtein function described in the slides uses an array to store intermediate results (matrix M on slide 33). Implement a recursive version of Levenshtein distance (should not use any auxiliary array).
- Test and time your recursive function with strings of increasing length.

Exercise 2

- Modify your recursive Levenshtein function so that it caches the results it computes in a dictionary.
- Test and time your modified recursive function with strings of increasing length.
- Were the modifications worth the effort?

Exercise 3

- Write a non-recursive function to compute the DP table.
- Write a function that lists the optimal sequence of edit actions from a DP table.