'INF 4130 Exercises, Sept. 18 and 20, 2018

Exercise 1

1.1 Run the edit distance algorithm (on paper) with two similar words, e.g., "algori" og "logari", and with two identical words.

1.2 Show how to implement the algorithm using only one column (or row) plus a few additional variables.

1.3 Solve the problem given in the last sentence of section 20.5 on page 645. That is: In the slides we originally wanted to find an algorithm for searching through a string T, and look for substrings S = T[p], T[p+1], … , T[q] of T similar to a given string P. We can assume that we want to find the first substring of T whose edit distance to P is less than or equal to a given K (or report that no such substring occurs).

Exercise 2

Look into memoization – using a table as in standard dynamic programming, but with an algorithm following the recursive formula top-down. The trick is now that each recursive call first looks into the table, and checks if the answer to the current sub-problem is already calculated. If it is, this value is used, otherwise we have to do recursive calls to solve the necessary smaller problems.

Write such an algorithm for finding the edit distance between two strings P and T.

Exercise 3 (repetition and extension of a problem from the lecture)

We assume that we have an integer array "C[0:m-1, 0:n-1]" filled with positive intergers. We want to find the cheapest path from C[0,0] to C[m-1, n-1], where the cost of a path is the sum of the integers in the array-entries it passes through. A path can only pass from one entry C[i,j] to the one to the right of it (C[i, j+1]), to the one below it (C[i+1, j]) or to the one diagonally down to the right of it (C[i+1,j+1]), and it has to stay within the borders of the array.

3.1 Describe what sort of table you want to use for solving this problem with dynamic programming

3.2 Sketch a program for solving the problem

3.3 This problem can be seen as a shortest path problem with (m x n) nodes, and is thereby solvable with Dijkstra's algorithm. Compare the speed of the two methods, and try to find the source of the difference.

3.4 Assume we want to solve the above problem with top-down recursive memorization. In what cases can we avoid computing all the entries in our table?

Exercise 4

Make sure that everybody understand why the Fibonacci algorithm we looked in the lecture (see slides) is *not* a polynomial time algorithm, but that it uses exponential time in the size of the input (the number of digits in n). Compare with the algorithms for addition and multiplication of two integers, which indeed are polynomial time algorithms in the size (number of digits) of the problem.