University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
P. Schneider

Algorithms and Data Structures
Winter Term 2019/2020
Exercise Sheet 10

Remark: For this exercise, please watch the 14^{th} (final) video lecture.

Exercise 1: Topics for Final Lesson

In our final session (29th of January) we will have time to repeat some of the topics that you had difficulties with. For this purpose please send me an email (philipp.schneider@cs.uni-freiburg.de) with the topic "AD VOTE" and provide a list of three topics that you would like to repeat. From the topics most wished for I will compile the final, 11^{th} exercise sheet.

Exercise 2: Edit Distance

Let $A = a_1 \dots a_n, B = b_1 \dots b_m$ be two words. For $k \leq n, \ell \leq m$ let $A_k = a_1 \dots a_k, B_\ell = b_1 \dots b_\ell$ be the prefixes of A und B. Let $ED_{k,\ell} := ED(A_k, B_\ell)$ be the edit distance of A_k, B_ℓ. Use the dynamic programming algorithm from the lecture to compute $ED_{n,m}$ for the inputs $A = TORRENT$ und $B = RODENT$ by filling a table with values $ED_{k,\ell}$.

Exercise 3: Binomial Coefficient

Consider the following recursive definition of the binomial coefficient

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1},$$

with base cases $\binom{n}{0} = \binom{n}{n} = 1$. Give an algorithm that uses the principle of dynamic programming to compute $\binom{n}{k}$ in $\mathcal{O}(n \cdot k)$ time steps. Argue the running time of your algorithm

Exercise 4: Computing Minimum Change

Assume you are a vending machine and need to output an amount $N \in \mathbb{N}$ using coins with denominations $c_1, \dots, c_n \in \mathbb{N}$ of which you have an unlimited supply. To make things simpler you do not actually have to compute the minimum cardinality set of coins that make up the amount N, but only the size of such a set (if it exists). Give an algorithm with runtime $\mathcal{O}(n \cdot N)$ that uses the principle of dynamic programming to compute the number of coins required to return an amount N, or ∞ if the amount can *not* be written as a weighted sum of $c_1, \dots, c_n$. Argue the runtime of your algorithm.