Institut for Matematik og Datalogi
Syddansk Universitet, Odense

24. April 2025

Exercise Sheet Week 10

DSK814

A. Solve in the practice sessions

1. Cormen et al., 4. Edition: Exercise 14.1-3 (page 373) [Cormen et al., 3. Edition: Exercise 15.1-3 (page 370)].

   Consider a modification of the rod-cutting problem in which, in addition to a price p_i for each rod, each cut incurs a fixed cost of c. The revenue associated with a solution is now the sum of the prices of the pieces minus the costs of making the cuts. Give a dynamic-programming algorithm to solve this modifed problem.

2. Cormen et al., 4. Edition: Exercise 14.4-1 (page 399) [Cormen et al., 3. Edition: Exercise 15.4-1 (page 396)].

   Determine an LCS of $\langle 1, 0, 0, 1, 0, 1, 0, 1 \rangle$ and $\langle 0, 1, 0, 1, 1, 0, 1, 1, 0 \rangle$.

3. Cormen et al., 4. Edition: Exercise 14.4-2 (page 399) [Cormen et al., 3. Edition: Exercise 15.4-2 (page 396)].

   Give pseudocode to reconstruct an LCS from the completed c table and the original sequences $X = \langle x_1, x_2, \dots, x_m \rangle$ and $Y = \langle y_1, y_2, \dots, y_n \rangle$ in $O(m + n)$ time, without using the b table.

4. Find the largest weighted subsequence of two character sequences X and Y, i.e. the sum of the weights of the characters in the subsequence of the sequences X and Y. Using the table below, we can calculate the weight of the subsequence $Z = acd$ of the sequences $X = acbd$ and $Y = bacda$ by $2 + 1 + 3 = 6$.

   a) Find the largest weighted subsequence for the sequences $X = acbd$ and $Y = bacda$, i.e. in calculating $W(n, m)$.

Sign c	a	b	c	d
Weight $w(c)$	2	4	1	3

b) Provide the pseudocode of a dynamic programming algorithm calculating the largest weight of a subsequence of X and Y.

B(I). Solve at home before tutorial in week 11

1. (∗) Cormen et al., 4. Edition: Exercise 4.2-5 (page 90) [Cormen et al., 3. Edition: Exercise 4.2-7 (page 83)].

   Show how to multiply the complex numbers $a + bi$ and $c + di$ using only three multiplications of real numbers. The algorithm should take a, b, c and d as input and produce the real component $ac - bd$ and the imaginary component $ad - bc$ separately.

   Hint: The product of two complex numbers $a + bi$ and $c + di$ is $(ac - bd) + (ad + bc)i$. Thus, the task is to calculate $ac - bd$ and $ad + bc$ from a, b, c and d using only three multiplications. The answer is somewhat similar to Strassen's algorithm but is much simpler.

B(II). Solve at home before tutorial in week 11

1. Cormen et al., 4. Edition: Exercise 14.4-5 (page 399) [Cormen et al., 3. Edition: Exercise 15.4-5 (page 397)].

   Give an $O(n^2)$-time algorithm to find the longest monotonically increasing subsequence of a sequence of n numbers.

   Hint: make a (1-dimensional) table for $l(i)$ with $i = 1, \ldots, n$, where $l(i)$ is the length of the longest monotonically increasing subsequence ending at the i-th number. Then solve this slightly modified problem with dynamic programming and use the table of solutions to solve the original problem.

   You can also find longest monotonically increasing subsequences by considering the input sequence as a string of numbers. Then you solve the LCS problem on the string itself and its sorted version such that all the numbers come in increasing order (challenge: prove that this solves the same problem) as we already have a dynamic programming solution. Is the running time the same or is it different?

2. Cormen et al., 4. Edition: problem 14-2 (page 407) [Cormen et al., 3. Edition: problem 15-2 (page 405)].

   A *palindrome* is a nonempty string over some alphabet that reads the same forward and backward. Examples of palindromes are all strings of length 1, `civic`, `racecar`, and `aibohphobia` (fear of palindromes). Give an efficient algorithm to find the longest palindrome that is a subsequence of a given input string. For example, given the input `character`, your algorithm should return `carac`. What is the running time of your algorithm?

   Hint: To solve it with dynamic programming, define the subproblems as the substrings $x_i x_{i+1} \ldots x_{j-1} x_j$ for $i \leq j$. In the analysis, consider both ends of the subproblem (i.e. of x_i and x_j) when you try to relate it to smaller subproblems. Additionally, the analysis is somewhat similar to that of Longest Common Subsequence.

   You can also solve this problem by solving the LCS problem for the string and its inverse (hard challenge: prove this) as we already have a dynamic programming solution. Is the running time the same or different?